aws

Guida per gli sviluppatori

Amazon DocumentDB



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

Amazon DocumentDB: Guida per gli sviluppatori

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

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

Table of Contents

Che cos'è Amazon DocumentDB 1
Panoramica di 1
Cluster 3
Istanze 4
Regioni e zone di disponibilità 8
Regioni 8
Zone di disponibilità 8
Prezzi 11
Prova gratuita 12
Monitoraggio 12
Interfacce 12
Console di gestione AWS 12
AWS CLI 13
Driver MongoDB 13
Fasi successive 13
Come funziona 13
Versioni di Amazon DocumentDB 14
Endpoint Amazon DocumentDB 16
Supporto TLS 20
Storage Amazon DocumentDB 20
Replica Amazon DocumentDB 21
Affidabilità di Amazon DocumentDB 22
Leggi le opzioni di preferenza 23
Il TTL elimina 27
Risorse fatturabili 28
Cos'è un database di documenti? 31
Casi d’uso 32
Comprendere i documenti 33
Utilizzo dei documenti 38
Guida introduttiva 51
Prerequisiti 51
Fase 1: creazione di un cluster 53
Fase 2: Connettiti al tuo cluster 56
Fase 3: Inserimento ed interrogazione dei dati 57

Fase 4: Esplora ... 59
Avvio rapido all'utilizzo CloudFormation ... 59
Prerequisiti ... 60
Avvio di un Amazon DocumentDB CloudFormation stack ... 61
Accesso al cluster Amazon DocumentDB ... 66
Protezione dalla terminazione e protezione dall'eliminazione ... 67
Migrazione e aggiornamento ... 68
Guida introduttiva rapida ... 68
Prepara la fonte DMS ... 69
Configura DMS ... 69
Abilita la compressione DocumentDB ... 69
Crea un'attività di replica ... 69
Monitoraggio dell'avanzamento ... 70
Informazioni aggiuntive ... 70
Runbook sulla migrazione ... 71
Compatibilità ... 72
Rilevamento dei carichi di lavoro ... 74
Crea un nuovo cluster di destinazione ... 78
Migrazione dell'indice ... 78
Migrazione degli utenti ... 79
Migrazione dei dati ... 81
Monitoraggio ... 92
Convalida ... 94
Migrazione da Couchbase Server ... 100
Introduzione ... 100
Confronto con Amazon DocumentDB ... 100
Individuazione ... 103
Pianificazione ... 108
Migrazione ... 111
Convalida ... 120
Date di supporto delle versioni del motore ... 123
Calendario di rilascio per le versioni principali di Amazon DocumentDB ... 123
Calendario di rilascio per le versioni secondarie di Amazon DocumentDB ... 124
Aggiornamento di una versione principale del motore ... 125
Percorsi di aggiornamento supportati ... 126
Prerequisiti ... 127

Best practice ... 129
Cosa cambia dopo l'aggiornamento ad Amazon DocumentDB 8.0 ... 132
Post-upgrade considerazioni per i cluster aggiornati dalla versione 3.6 o 4.0 ... 133
Esecuzione dell'aggiornamento ... 134
Risoluzione dei problemi ... 136
Aggiornamento di una versione secondaria del motore ... 136
Operazioni preliminari all'aggiornamento a una versione secondaria ... 137
Esecuzione di un aggiornamento a una versione secondaria ... 137
Verifica dell'aggiornamento della versione secondaria ... 138
Aggiornamento tramite AWS DMS ... 139
Fase 1: abilitare i flussi di modifica ... 139
Passaggio 2: modifica la durata di conservazione dei flussi di modifica ... 140
Fase 3: Esegui la migrazione degli indici ... 140
Fase 4: Creare un AWS DMS istanza di replica ... 141
Fase 5: Creare un AWS DMS endpoint di origine ... 143
Fase 6: Creare un AWS DMS endpoint di destinazione ... 146
Fase 7: Creare ed eseguire un'attività di migrazione ... 148
Fase 8: Modifica dell'endpoint dell'applicazione nel cluster Amazon DocumentDB di destinazione ... 150
Uso di versioni con supporto a lungo termine (Long-Term Support, LTS) ... 150
Versione LTS attuale ... 150
Chi dovrebbe usare LTS ... 151
Come creare un nuovo cluster LTS ... 151
Come eseguire l'aggiornamento dalla versione 3.6.0 o 4.0.0 del motore a un cluster LTS 5.0.0 ... 152
Come verificare se il cluster della versione 5.0.0 del mio motore esistente è su LTS ... 152
Supporto esteso ... 153
Panoramica del supporto esteso ... 154
Costi del supporto esteso ... 154
Responsabilità ... 155
Caratteristiche e configurazioni ... 156
Stato del motore ... 156
Tipi di cluster ... 156
Funzionalità ... 157
Tipi di istanza ... 158
AWS integrazioni ... 159

Scelta di regioni e zone di disponibilità 160
Disponibilità nelle regioni 161
OpenSearch integrazione 164
Amazon OpenSearch Service come destinazione 164
Limitazioni 168
Sviluppando 169
Connect a DocumentDB 169
Connessione a livello di codice 170
Connessione dall'esterno di un Amazon VPC 200
Connessione come set di repliche 202
Connect tramite Studio 3T 207
Connect usando DataGrip 216
Connettiti usando Amazon EC2 221
Connettiti utilizzando il driver JDBC 259
Connettiti utilizzando il driver ODBC 275
Programma con Amazon DocumentDB 289
Guida alla programmazione Java DocumentDB 289
Utilizzo della convalida dello schema JSON 322
Esecuzione di query 332
Interrogazione di documenti 332
Piano di query 335
Spiegare i risultati 338
Pianificatore di query v2 343
Pianificatore di query v3 358
Dati geospaziali 363
Indice parziale 369
Ricerca testuale 376
Indici 384
gestione degli indici 385
Tipi di indice 396
Proprietà dell'indice 405
Utilizzo dei flussi di modifica 424
Operazioni supportate 425
Eventi estesi 426
Fatturazione 426
Limitazioni 427

Abilitazione dei flussi di modifica 428
Utilizzo dei flussi di modifica con Python 430
Ricerca completa del documento 432
Usare $change StreamSplitLargeEvent per dividere eventi di grandi dimensioni in frammenti 433
Ripresa di un flusso di modifiche 434
Ripresa con `startAtOperationTime` 435
Ripresa con `postBatchResumeToken` 437
Transazioni 439
Modifica della durata di conservazione del registro 439
Flussi di modifica sulle istanze secondarie 442
Transazioni 444
Requisiti 444
Best practice 444
Limitazioni 445
Monitoraggio e diagnostica 446
Livello di isolamento delle transazioni 446
Casi d'uso 447
Comandi supportati 484
Funzionalità non supportate 485
Sessioni 486
Errori di transazione 487
Retryable scrive 488
Requisiti 488
Operazioni supportate 488
Limitazioni 489
Abilitazione 490
Come funziona 490
Gestione degli errori 490
Migrazione 491
Utilizzo della collazione 491
Limitazioni 492
Utilizzo delle viste di 492
Best practice 493
Compatibilità con gli operatori di aggregazione 494
Sfruttando indici e visualizzazioni 494

IA generativa 494
SageMaker Tela 495
Ricerca vettoriale 497
Utilizzo AWS Lambda con flussi di modifica 511
Limitazioni 511
Compatibilità con MongoDB 513
Compatibilità con MongoDB 8.0 513
Cosa c'è di nuovo in Amazon DocumentDB 8.0 513
Inizia a usare Amazon DocumentDB 514
Aggiornamento o migrazione ad Amazon DocumentDB 5.0 o 8.0 515
Differenze funzionali 515
Compatibilità con MongoDB 5.0 515
Cosa c'è di nuovo in Amazon DocumentDB 5.0 516
Cosa c'è di nuovo in Amazon DocumentDB 5.0.1 516
Inizia a usare Amazon DocumentDB 5.0 514
Aggiornamento o migrazione ad Amazon DocumentDB 5.0 515
Differenze funzionali 515
Compatibilità con MongoDB 4.0 519
Caratteristiche di Amazon DocumentDB 4.0 520
Inizia a usare Amazon DocumentDB 4.0 521
Aggiornamento o migrazione ad Amazon DocumentDB 4.0 521
Differenze funzionali 522
Differenze funzionali con MongoDB 523
Vantaggi funzionali di Amazon DocumentDB 523
Differenze funzionali aggiornate 524
Differenze funzionali con MongoDB 528
API, operazioni e tipi di dati MongoDB supportati 537
Comandi del database 538
Operatori di interrogazione e proiezione 548
Operatori di aggiornamento 551
Dati geospaziali 553
Metodi Cursor 555
Operatori di pipeline di aggregazione 557
Tipi di dati 575
Indici e proprietà degli indici 576
Operatori di pipeline di aggregazione 578

Dati geospaziali ... 1212
Operatori di interrogazione e proiezione ... 1236
Aggiorna gli operatori ... 1305
Gestione di cluster e istanze ... 1360
Panoramica delle attività operative ... 1360
Aggiungere una replica a un cluster Amazon DocumentDB ... 1361
Descrizione di cluster e istanze ... 1361
Creazione di un'istantanea del cluster ... 1364
Ripristino da uno snapshot ... 1365
Rimuovere un'istanza da un cluster ... 1366
Eliminazione di un cluster ... 1366
Cluster globali ... 1367
Cos'è un cluster globale? ... 1367
In che modo sono utili i cluster globali? ... 1367
Quali sono le attuali limitazioni dei cluster globali? ... 1368
Guida introduttiva rapida ... 1369
Gestire i cluster globali ... 1381
Connessione di cluster globali ... 1388
Monitoraggio dei cluster globali ... 1389
Ripristino di emergenza ... 1390
Gestione dei cluster ... 1415
Comprendere i cluster ... 1415
Impostazioni del cluster ... 1418
Configurazioni di storage in cluster ... 1421
Determinazione dello stato di un cluster ... 1424
Ciclo di vita del cluster ... 1425
Scalabilità dei cluster ... 1469
Clonazione di un volume per un cluster ... 1472
Comprendere la tolleranza ai guasti del cluster ... 1485
Gestione di istanze ... 1487
Determinazione dello stato di un'istanza ... 1487
Ciclo di vita dell'istanza ... 1487
Gestione delle classi di istanze ... 1511
NVMe-backed istanze ... 1524
Gestione dei gruppi di sottoreti ... 1527
Creazione di un gruppo di sottoreti ... 1528

Descrivere un gruppo di sottoreti ... 1533
Modifica di un gruppo di sottoreti ... 1536
Eliminazione di un gruppo di sottoreti ... 1539
Disponibilità e replica elevate ... 1541
Dimensionamento della lettura ... 1541
Elevata disponibilità ... 1541
Aggiunta di repliche di ... 1543
Failover ... 1543
Ritardo di replica ... 1548
Gestione degli eventi ... 1550
Visualizzazione delle categorie di eventi ... 1550
Visualizzazione degli eventi di Amazon DocumentDB ... 1553
Gestione dei gruppi di parametri del cluster ... 1555
Descrizione dei gruppi di parametri del cluster ... 1556
Creazione di gruppi di parametri del cluster ... 1564
Modifica dei gruppi di parametri del cluster ... 1567
Modifica dei cluster per utilizzare gruppi di parametri di cluster personalizzati ... 1572
Copia dei gruppi di parametri del cluster ... 1573
Reimpostazione dei gruppi di parametri del cluster ... 1576
Eliminazione dei gruppi di parametri del cluster ... 1578
Riferimento ai parametri del cluster ... 1581
Comprendere gli endpoint ... 1604
Individuazione degli endpoint di un cluster ... 1604
Individuazione dell'endpoint di un'istanza ... 1607
Connessione agli endpoint ... 1610
Informazioni sugli ARN di Amazon DocumentDB ... 1612
Costruzione di un ARN ... 1612
Ricerca di un ARN ... 1616
Applicazione di tag alle risorse ... 1618
Panoramica sui tag delle risorse di ... 1619
Vincoli relativi ai tag ... 1620
Aggiunta o aggiornamento dei tag ... 1620
Come elencare i tag ... 1622
Rimozione dei tag ... 1623
Manutenzione di Amazon DocumentDB ... 1625
Azioni di manutenzione per Amazon DocumentDB ... 1626

Numerazione delle versioni 1627
Gestisci le finestre di manutenzione 1628
Notifiche delle patch del motore 1630
Visualizza la manutenzione in sospeso 1632
Aggiornamenti del motore 1635
Versioni secondarie 1639
Aggiornamenti del sistema operativo 1639
User-initiated aggiornamenti 1644
Cluster globali 1645
Comprendere i ruoli collegati ai servizi 1646
Service-linked autorizzazioni di ruolo 1646
Creazione di un ruolo collegato ai servizi 1648
Modifica di un ruolo collegato al servizio 1648
Eliminazione del ruolo collegato ai servizi 1649
Regioni supportate per i ruoli collegati al servizio Amazon DocumentDB 1650
Ottimizzazione delle prestazioni 1651
Problemi di prestazioni comuni 1651
CPU elevata 1651
Problemi di connessione 1654
Query in esecuzione lenta 1661
Anti-patterns 1665
Indice composto anti-pattern con più di 3 attributi 1665
Interrogazioni di lunga durata 1668
Indici poco utilizzati e ridondanti 1670
Multi-key indici con matrici di grandi dimensioni 1673
Suggerimenti per il miglioramento delle prestazioni 1674
1. Usa $match come prima fase nelle pipeline di aggregazione 1674
2. Usa $project in Aggregation Pipeline per ridurre al minimo le dimensioni dei dati della pipeline 1675
3. Abilita la compressione dei documenti per ridurre i costi e i costi di archiviazione e migliorare le prestazioni delle query I/O 1675
4. Sfrutta gli indici per prestazioni di query ottimali 1676
5. Ottimizza i modelli di dati in base ai modelli di query 1677
Analisi del piano di interrogazione 1679
Tempistica complessiva 1681
fasi di esecuzione 1681

Comprensione dei documenti esaminati in Explain Plan Execution Stats 1683
Index-only scansioni 1691
Quando Amazon DocumentDB utilizza una scansione solo indicizzata 1692
Requisiti per una domanda coperta 1694
Restrizioni 1694
Conferma di una scansione solo indicizzata 1695
Best practice 1696
Pianifica i filtri della cache 1697
Comandi supportati 1698
Forme di interrogazione supportate 1699
Impostazione, elenco e cancellazione dei filtri 1700
Verifica dell'applicazione di un filtro 1702
Esempi 1703
Comportamento con suggerimenti e modifiche all'indice 1705
Best practice 1706
Linee guida operative di base 1707
Dimensionamento delle istanze 1708
Utilizzo degli indici 1710
Best practice di sicurezza 1711
Ottimizzazione dei costi 1711
Utilizzo di parametri per identificare problemi a livello di prestazioni 1712
Carichi di lavoro TTL e serie temporali 1716
Migrazioni 1717
Lavorare con i gruppi di parametri del cluster 1717
Interrogazioni sulla pipeline di aggregazione 1717
`BatchInsert e BatchUpdate` 1718
Gestione degli indici 1718
Creazione di indici Amazon DocumentDB 1718
Gestione degli indici Amazon DocumentDB 1725
Gestione della compressione dei documenti 1727
Gestione della compressione dei documenti 1728
Monitoraggio della compressione dei documenti 1729
Gestione della compressione basata su dizionari 1730
Considerazioni sulle prestazioni 1731
Attivazione della compressione basata su dizionari 1732
Nozioni di base 1733

Monitoraggio 1733
Sicurezza 1735
Gestione delle password con AWS Secrets Manager 1736
Limitazioni per l'integrazione di Secrets Manager con Amazon DocumentDB 1737
Panoramica sulla gestione delle password degli utenti primari con AWS Secrets Manager 1737
Applicazione della gestione Amazon DocumentDB della password utente principale in AWS Secrets Manager 1738
Gestione della password utente principale per un cluster con Secrets Manager 1738
Protezione dei dati 1739
Client-side crittografia a livello di campo 1740
Crittografia dei dati a riposo 1747
Crittografia dei dati in transito 1755
Gestione delle chiavi 1766
Identity and Access Management 1767
Destinatari 1767
Autenticazione con identità 1768
Gestione dell’accesso tramite policy 1769
Come funziona Amazon DocumentDB con IAM 1771
Identity-based esempi di policy 1777
Risoluzione dei problemi 1781
Gestione delle autorizzazioni di accesso alle tue risorse Amazon DocumentDB 1783
Utilizzo di policy basate su identità (policy IAM) 1788
AWS politiche gestite per Amazon DocumentDB 1792
Riferimento alle autorizzazioni API Amazon DocumentDB 1812
Cross-service prevenzione sostitutiva confusa 1821
Autenticazione tramite identità IAM 1822
Guida introduttiva agli utenti e ai ruoli IAM 1823
Configurazione dei tipi di calcolo 1826
Monitoraggio delle richieste di autenticazione 1827
Utilizzo dell'autenticazione IAM 1827
Driver che supportano IAM 1828
Domande frequenti sull'autenticazione dell'identità IAM 1828
Autenticazione tramite SCRAM-SHA-256 1829
Creazione di utenti 1829
Aggiornamento degli utenti 1830
Autenticazione 1831

Gestione degli utenti di Amazon DocumentDB 1831
Utente primario e serviceadmin 1832
Creazione di utenti aggiuntivi 1832
Password a rotazione automatica 1834
Role-Based Controllo degli accessi 1835
Concetti RBAC 1836
Guida introduttiva ai ruoli integrati in RBAC 1838
Guida introduttiva ai ruoli definiti dall'utente RBAC 1841
Connessione ad Amazon DocumentDB come utente 1846
Comandi comuni 1847
Differenze funzionali 1852
Limits 1853
Accesso al database tramite Role-Based Access Control 1853
Aggiornamento dei certificati 1862
Aggiornamento dell'applicazione e del cluster Amazon DocumentDB 1862
Rotazione automatica dei certificati del server 1867
Risoluzione dei problemi 1868
Domande frequenti 1868
Convalida della conformità 1875
Resilienza 1876
Sicurezza dell’infrastruttura 1877
Endpoint VPC (AWS PrivateLink) 1878
Considerazioni sugli endpoint VPC dell' 1879
Disponibilità nelle regioni 1879
Creazione di un endpoint VPC di interfaccia per l'API Amazon DocumentDB 1880
Creazione di una policy sugli endpoint VPC per l'API Amazon DocumentDB 1880
Best practice di sicurezza 1882
Audit degli eventi 1883
Eventi supportati 1884
Abilita il controllo 1889
Disabilitare il controllo 1896
Accedi ai tuoi eventi 1899
Filtrare gli eventi DML 1899
Amazon VPC e Amazon DocumentDB 1905
Cluster DocumentDB in un VPC 1906
Accesso a un cluster in un VPC 1918

Crea un IPv4-only VPC per un cluster ... 1920
Crea un VPC dual-stack per un cluster ... 1927
Backup e ripristino ... 1938
Backup e ripristino: concetti ... 1939
Informazioni sull'utilizzo dello storage di backup ... 1941
Dumping, ripristino, importazione ed esportazione dei dati ... 1943
mongodump ... 1943
mongorestore ... 1944
mongoexport ... 1945
mongoimport ... 1945
Tutorial ... 1946
Considerazioni sulle istantanee del cluster ... 1948
Storage di backup ... 1949
Backup window (Finestra di backup) ... 1950
Periodo di conservazione dei backup ... 1952
Copia la crittografia delle istantanee del cluster ... 1952
Confronto tra istantanee automatiche e manuali ... 1953
Creazione di un'istantanea manuale del cluster ... 1955
Copiare un'istantanea del cluster ... 1958
Copia di snapshot condivise ... 1959
Copiare istantanee tra Regioni AWS ... 1959
Limitazioni ... 1959
Gestione della crittografia ... 1959
Considerazioni sui gruppi di parametri ... 1960
Copiare un'istantanea del cluster ... 1960
Condivisione di uno snapshot di un cluster ... 1970
Condivisione di uno snapshot crittografato ... 1971
Condivisione di uno snapshot ... 1974
Ripristino da un'istantanea del cluster ... 1977
Ripristino a un determinato punto nel tempo ... 1985
Eliminazione di un'istantanea del cluster ... 1991
Monitoraggio e registrazione dei log ... 1994
Monitoraggio dello stato di un cluster ... 1995
Valori di stato del cluster ... 1996
Monitoraggio dello stato di un cluster ... 1997
Monitoraggio dello stato di un'istanza ... 1999

Valori di stato delle istanze 2000
Monitoraggio dello stato dell'istanza utilizzando Console di gestione AWS or AWS CLI 2002
Valori dello stato di integrità dell'istanza 2004
Monitoraggio dello stato di integrità dell'istanza utilizzando Console di gestione AWS 2004
Visualizzazione dei consigli di Amazon DocumentDB 2006
Abbonamenti a eventi 2009
Sottoscrizione agli eventi 2010
Gestione delle sottoscrizioni a 2012
Categorie e messaggi 2016
Monitoraggio di Amazon DocumentDB con CloudWatch 2019
Metriche di Amazon DocumentDB 2019
Visualizzazione CloudWatch dei dati 2038
Dimensioni di Amazon DocumentDB 2044
Monitoraggio delle metriche opcounter 2044
Monitoraggio delle connessioni al database 2044
Registrazione delle chiamate API di Amazon DocumentDB con CloudTrail 2045
Informazioni su Amazon DocumentDB in CloudTrail 2045
Operazioni di profilazione 2046
Operazioni supportate 2047
Limitazioni 2048
Abilitazione del profiler 2048
Disattivazione del profiler 2053
Disattivazione dell'esportazione dei log del profiler 2053
Accesso ai log del tuo profiler 2056
Domande comuni 2056
Monitoraggio con Performance Insights 2057
Concetti relativi a Performan 2058
Abilitazione e disattivazione di Performance Insights 2062
Configurazione delle policy di accesso per Performance Insights 2065
Analisi delle metriche utilizzando il pannello di controllo Performance Insights 2069
Recupero dei parametri con l'API Performance Insights 2088
CloudWatch Metriche di Amazon per Performance Insights 2103
Approfondimenti sulle prestazioni per le contrometriche 2105
Serverless 2108
Casi d'uso Serverless 2108

Utilizzo di Amazon DocumentDB serverless per i carichi di lavoro con provisioning esistenti ... 2110
Vantaggi di Amazon DocumentDB serverless ... 2110
Come funziona il serverless ... 2112
Panoramica di ... 2112
Configurazioni per i cluster Amazon DocumentDB ... 2113
Capacità di scalabilità serverless di Amazon DocumentDB ... 2114
Scalabilità senza server di Amazon DocumentDB ... 2116
Stato inattivo (0,5 DCU) ... 2118
Requisiti e limitazioni serverless ... 2118
Requisiti delle funzionalità ... 2118
Creazione di un cluster che utilizza serverless ... 2120
Creazione di un cluster serverless Amazon DocumentDB ... 2120
Aggiungere un'istanza serverless di Amazon DocumentDB ... 2123
Migrazione verso serverless ... 2124
Migrazione dei cluster DocumentDB esistenti a quelli serverless ... 2124
Migrazione da MongoDB a DocumentDB serverless ... 2129
Gestione senza server ... 2129
Visualizzazione e modifica della configurazione dell'intervallo di capacità di scalabilità di un cluster ... 2129
Visualizzazione e modifica del livello di promozione dei lettori serverless ... 2133
Limiti delle istanze serverless ... 2135
Configurazione di scalabilità senza server ... 2137
Scelta dell'intervallo di capacità di scalabilità per un cluster serverless DocumentDB ... 2138
Scelta dell'`MinCapacity`impostazione per un cluster serverless DocumentDB ... 2138
Scelta dell'`MaxCapacity`impostazione per un cluster serverless DocumentDB ... 2140
Esclusione di errori dovuti all'esaurimento della memoria ... 2143
Perché la mia istanza serverless non viene ridimensionata? ... 2144
Monitoraggio senza server ... 2145
Memoria insufficiente: stato dei parametri incompatibili ... 2145
Metriche Amazon per DocumentDB serverless CloudWatch ... 2145
Monitoraggio delle prestazioni serverless di DocumentDB con Performance Insights ... 2150
Cluster elastici ... 2151
Casi d'uso di Elastic Cluster ... 2152
Profili utente ... 2152
Gestione dei contenuti e record storici ... 2152

Vantaggi dei cluster elastici 2152
AWS integrazione dei servizi 2152
Disponibilità di regioni e versioni 2153
Disponibilità delle regioni per i cluster elastici 2153
Disponibilità delle versioni 2154
Limitazioni 2154
Gestione elastica dei cluster 2154
Operazioni di interrogazione e scrittura 2155
Gestione della raccolta e dell'indice 2155
Amministrazione e diagnostica 2156
Opt-in features 2156
Come funziona 2156
Sharding elastico dei cluster Amazon DocumentDB 2157
Migrazione elastica dei cluster 2160
Scalabilità elastica dei cluster 2160
Affidabilità del cluster elastico 2160
Storage e disponibilità dei cluster elastici 2160
Differenze funzionali tra Amazon DocumentDB 4.0 e i cluster elastici 2161
Nozioni di base 2162
Prerequisiti 2163
Fase 1: Creare un cluster elastico 2164
Passaggio 2: connettiti al tuo cluster elastico 2170
Passaggio 3: condividi la tua raccolta, inserisci ed interroga i dati 2171
Fase 4: Esplora 2175
Best practice 2175
Scelta di chiavi condivise 2175
Gestione delle connessioni 2176
Raccolte non condivise 2176
Scalabilità dei cluster elastici 2176
Monitoraggio dei cluster elastici 2177
Gestione dei cluster elastici 2177
Modifica delle configurazioni dei cluster elastici 2178
Monitoraggio di un cluster elastico 2181
Eliminazione di un cluster elastico 2185
Gestione delle istantanee dei cluster elastici 2187
Arresto e avvio di un cluster elastico 2201

Manutenzione dei cluster elastici 2206
Crittografia dei dati a riposo 2214
In che modo i cluster elastici Amazon DocumentDB utilizzano le sovvenzioni in AWS KMS 2216
Creazione di una chiave gestita dal cliente 2216
Monitoraggio delle chiavi di crittografia per i cluster elastici di Amazon DocumentDB 2218
Ulteriori informazioni 2223
Service-linked ruoli 2224
Service-linked autorizzazioni di ruolo per i cluster elastici 2224
Quote 2228
Tipi di istanze supportati 2228
Regioni supportate 2230
Quote regionali 2232
Quote di aggregazione 2235
Quote del cluster 2235
Quote delle istanze 2237
Vincoli per la denominazione 2244
Vincoli TTL 2246
Quote elastiche dei cluster 2246
Quote di frammenti di cluster elastici 2248
Quote di CPU, memoria, connessione e cursore del cluster elastico per shard 2248
Risoluzione dei problemi 2250
Problemi di connettività 2250
Impossibile connettersi a un endpoint Amazon DocumentDB 2250
Test di una connessione a un'istanza Amazon DocumentDB 2256
Connessione a un endpoint non valido 2256
La configurazione del driver influisce sul numero di connessioni 2257
Indici 2257
La creazione dell'indice non riesce 2257
L'indice in background genera problemi di latenza e fallisce 2258
Index bloat 2258
Prestazioni e utilizzo delle risorse 2260
Visualizza le statistiche di inserimento, aggiornamento ed eliminazione 2261
Analizza le prestazioni della cache 2262
Trova e termina le query di lunga durata o bloccate 2263
Visualizza un piano di interrogazione e ottimizza un'interrogazione 2265
Come posso visualizzare un piano di query nei cluster elastici? 2267

Elenca tutte le operazioni in esecuzione su un'istanza 2269
Scopri quando una query sta facendo progressi 2272
Determina il motivo per cui un sistema funziona improvvisamente lentamente 2275
Determinare la causa dell'utilizzo elevato della CPU 2277
Trova i cursori aperti su un'istanza 2278
Visualizza la versione attuale del motore Amazon DocumentDB 2278
Analizza l'utilizzo degli indici e identifica gli indici non utilizzati 2278
Identifica gli indici mancanti 2281
Come faccio a determinare il numero eccessivo di raccolte di database? 2282
Riepilogo delle domande utili 2284
Raccolta dei rifiuti 2285
Informazioni sulla Garbage Collection in Amazon DocumentDB 2285
Processo di raccolta dei rifiuti 2286
Architettura di archiviazione e archiviazione estesa 2287
Monitoraggio della raccolta dei rifiuti 2288
Esempio di output CollStats 2291
Domande frequenti 2293
Codici di errore 2296
Note di rilascio 2330
Note di rilascio della versione del motore Amazon DocumentDB 2331
Note di rilascio generali di Amazon DocumentDB 2395
Note sulla versione dei cluster elastici Amazon DocumentDB 2400
Cronologia dei documenti 2404
..... mmcdxvii

Cos'è Amazon DocumentDB (con compatibilità con MongoDB)

Amazon DocumentDB (con compatibilità con MongoDB) è un servizio di database veloce, affidabile e completamente gestito. Amazon DocumentDB semplifica la configurazione, la gestione e la scalabilità MongoDB-compatible dei database nel cloud. Con Amazon DocumentDB, puoi eseguire lo stesso codice dell'applicazione e utilizzare gli stessi driver e strumenti che usi con MongoDB.

Prima di utilizzare Amazon DocumentDB, è necessario esaminare i concetti e le funzionalità descritti in. Come funziona Successivamente, completa la procedura in Guida introduttiva.

Argomenti

- Panoramica di Amazon DocumentDB
- Cluster
- Istanze
- Regioni e zone di disponibilità
- Prezzi di Amazon DocumentDB
- Monitoraggio
- Interfacce
- Fasi successive
- Amazon DocumentDB: come funziona
- Cos'è un database di documenti?

Panoramica di Amazon DocumentDB

Di seguito sono riportate alcune funzionalità di alto livello di Amazon DocumentDB:

- Amazon DocumentDB supporta due tipi di cluster: cluster basati su istanze e cluster elastici. I cluster elastici supportano carichi di lavoro con milioni di unità al secondo e petabyte di capacità di storage reads/writes . Per ulteriori informazioni sui cluster elastici, vedere. Utilizzo di Amazon DocumentDB Elastic Clusters Il contenuto seguente si riferisce ai cluster basati su istanze Amazon DocumentDB.
- Amazon DocumentDB aumenta automaticamente le dimensioni del volume di storage man mano che le esigenze di storage del database aumentano. Il volume di archiviazione aumenta con

incrementi di 10 GB, fino a un massimo di 256 TiB per la versione 8.0 e successive del motore (128 TiB per le versioni precedenti del motore). Non è necessario assegnare risorse di storage aggiuntive al cluster per far fronte alla crescita futura.

- Con Amazon DocumentDB, puoi aumentare il throughput di lettura per supportare richieste di applicazioni ad alto volume creando fino a 15 istanze di replica. Le repliche di Amazon DocumentDB condividono lo stesso storage sottostante, riducendo i costi ed evitando la necessità di eseguire scritture nei nodi di replica. Questa funzionalità libera più potenza di elaborazione per soddisfare le richieste di lettura e riduce il tempo di ritardo della replica, spesso fino a millisecondi a una cifra. È possibile aggiungere repliche in pochi minuti indipendentemente dalle dimensioni del volume di archiviazione. Amazon DocumentDB fornisce anche un endpoint di lettura, in modo che l'applicazione possa connettersi senza dover tenere traccia delle repliche man mano che vengono aggiunte e rimosse.
- Amazon DocumentDB consente di aumentare o ridurre le risorse di calcolo e memoria per ciascuna istanza. Le operazioni di dimensionamento delle risorse di calcolo in genere vengono completate in pochi minuti.
- Amazon DocumentDB viene eseguito in Amazon Virtual Private Cloud (Amazon VPC), quindi puoi isolare il database nella tua rete virtuale. Puoi anche configurare le impostazioni del firewall per controllare l'accesso di rete al cluster.
- Amazon DocumentDB monitora continuamente lo stato del tuo cluster. In caso di errore di un'istanza, Amazon DocumentDB riavvia automaticamente l'istanza e i processi associati. Amazon DocumentDB non richiede una ripetizione in caso di crash recovery dei redo log del database, il che riduce notevolmente i tempi di riavvio. Amazon DocumentDB isola inoltre la cache del database dal processo del database, consentendo alla cache di sopravvivere al riavvio di un'istanza.
- In caso di errore dell'istanza, Amazon DocumentDB automatizza il failover su una delle 15 repliche Amazon DocumentDB create in altre zone di disponibilità. Se non è stata eseguita alcuna replica e si verifica un errore, Amazon DocumentDB tenta di creare automaticamente una nuova istanza Amazon DocumentDB.
- La funzionalità di backup di Amazon DocumentDB consente il ripristino point-in-time per il tuo cluster. Questa caratteristica consente di ripristinare il cluster a qualsiasi momento compreso nel periodo di retention, fino agli ultimi 5 minuti. Puoi configurare il periodo di retention dei backup automatico fino a 35 giorni. I backup automatici sono archiviati in Amazon Simple Storage Service (Amazon S3), progettato per una durabilità del 99,9%. I backup di Amazon DocumentDB sono automatici, incrementali e continui e non hanno alcun impatto sulle prestazioni del cluster.

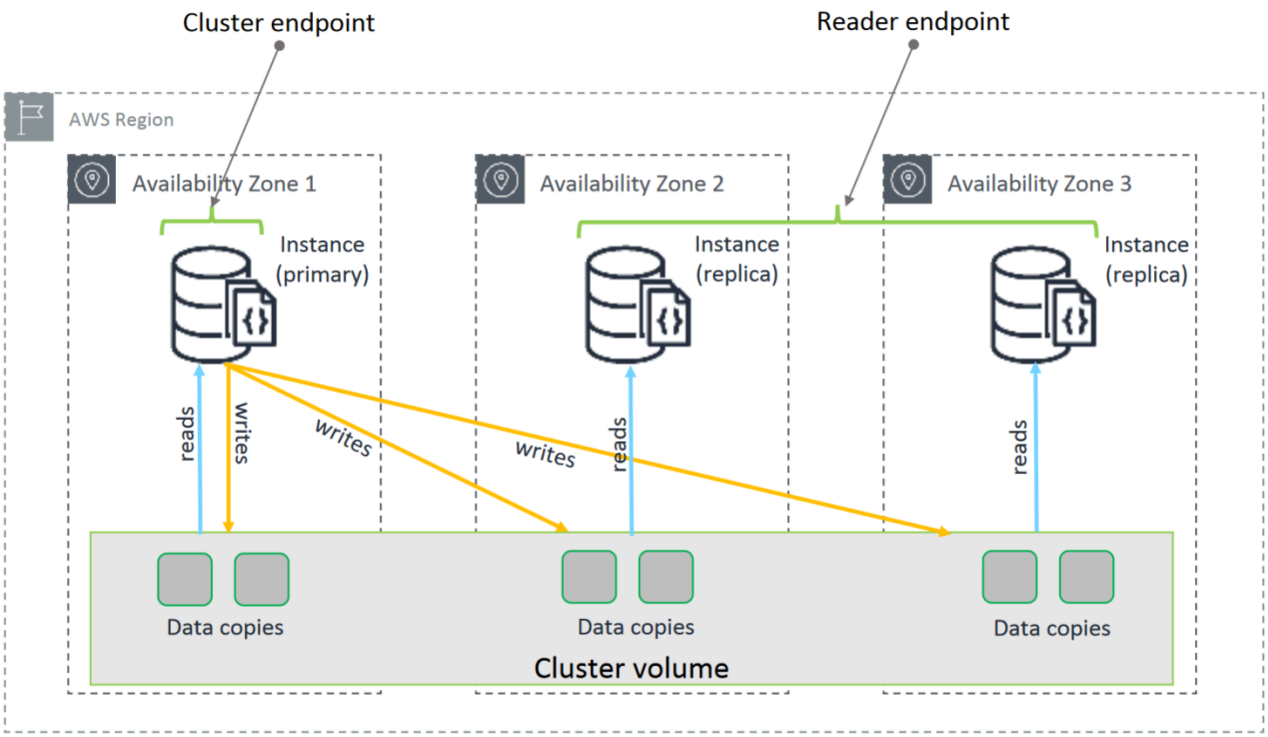
- Con Amazon DocumentDB, puoi crittografare i tuoi database utilizzando chiavi create e controllate tramite (). AWS Key Management Service AWS KMS In un cluster di database in esecuzione con la crittografia Amazon DocumentDB, i dati archiviati a riposo nello storage sottostante sono crittografati. Vengono crittografati anche i backup, le snapshot e le repliche automatici nello stesso cluster.
- Amazon DocumentDB è autorizzato ai sensi del Federal Risk and Authorization Management Program (FedRAMP). Ha l'autorizzazione FedRAMP High per le regioni AWS GovCloud (USA) e l'autorizzazione FedRAMP Moderate per le regioni degli Stati Uniti. AWS East/West Per dettagli AWS e iniziative in materia di conformità, vedi AWS Services in Scope by Compliance Program.

Se non conosci i AWS servizi, utilizza le seguenti risorse per saperne di più:

- AWS offre servizi per l'elaborazione, i database, l'archiviazione, l'analisi e altre funzionalità. Per una panoramica di tutti i AWS servizi, consulta Cloud Computing con Amazon Web Services.
- AWS fornisce una serie di servizi di database. Per indicazioni su quale servizio sia il migliore per il tuo ambiente, vedi Databases on AWS.

Cluster

Un cluster è composto da 0 a 16 istanze e da un volume di storage del cluster che gestisce i dati per tali istanze. Tutte le operazioni di scrittura vengono eseguite tramite l'istanza primaria. Tutte le istanze (primaria e repliche) supportano le operazioni di lettura. I dati del cluster vengono archiviati nel volume cluster e copiati in tre diverse zone di disponibilità.



I cluster basati su istanze Amazon DocumentDB 5.0 supportano due configurazioni di storage per un cluster di database: Amazon DocumentDB standard e Amazon DocumentDB. I/O-optimized Per ulteriori informazioni, consulta Configurazioni di storage del cluster Amazon DocumentDB.

Istanze

Un'istanza Amazon DocumentDB è un ambiente di database isolato nel cloud. Un'istanza può contenere più database creati dall'utente. Puoi creare e modificare un'istanza utilizzando Console di gestione AWS o. AWS CLI

La capacità di calcolo e di memoria di un'istanza sono determinate dalla relativa classe di istanza. Puoi selezionare l'istanza più adatta alle tue esigenze. Se le esigenze cambiano nel corso del tempo, puoi scegliere un'altra classe di istanza. Per le specifiche per la classe di istanza, consulta Specifiche della classe di istanza.

Le istanze Amazon DocumentDB vengono eseguite solo nell'ambiente Amazon VPC. Amazon VPC ti dà il controllo del tuo ambiente di rete virtuale: puoi scegliere il tuo intervallo di indirizzi IP, creare sottoreti e configurare elenchi di routing e controllo degli accessi (ACL).

Prima di poter creare istanze Amazon DocumentDB, devi creare un cluster per contenerle.

Non tutte le classi di istanze sono supportate in tutte. Regione AWS La tabella seguente mostra quali classi di istanze sono supportate in ciascuna regione.

> **Note**
>
> Per un elenco completo dei tipi di istanza supportati da Amazon DocumentDB in ogni classe di istanza, consultaSpecifiche della classe di istanza.

Classi di istanze supportate per regione

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Serverless
Stati Uniti orientali (Ohio)	Supportata	Supportato	Supportato	Supportato	Supportato	Supportato	Supportato	Supportata
Stati Uniti orientali (Virginia settentrionale)	Supportata	Supportato	Supportato	Supportato	Supportato	Supportato	Supportato	Supportata
Stati Uniti occidentali (Oregon)	Supportata	Supportato	Supportato	Supportato	Supportato	Supportato	Supportato	Supportata
Africa (Città del Capo)			Supportata	Supportato		Supportato	Supportato	Supportata
Sud America (San Paolo)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Hong Kong)	Supportata		Supportato	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Hyderabad)			Supportata	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Malesia)			Supportata			Supportato	Supportato	Supportata

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Serverless
Asia Pacifico (Mumbai)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Osaka)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Seul)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Sydney)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Giacarta)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Melbourne)			Supportata	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Singapore)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Thailandia)			Supportata			Supportato	Supportato	Supportata
Asia Pacifico (Tokyo)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Canada (Centrale)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Canada occidentale (Calgary)			Supportata	Supportato		Supportato	Supportato	Supportata
Europa (Francoforte)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Serverless
Europa (Zurigo)		Supportata	Supportato	Supportato		Supportato	Supportato	Supportata
Europa (Irlanda)	Supportata	Supportato	Supportato	Supportato	Supportato	Supportato	Supportato	Supportata
Europa (Londra)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Europa (Milano)			Supportata	Supportato		Supportato	Supportato	Supportata
Europa (Parigi)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Europa (Spagna)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Europa (Stoccolma)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Messico (centrale)			Supportata			Supportato	Supportato	Supportata
Medio Oriente (Emirati Arabi Uniti)			Supportata	Supportato		Supportato	Supportato	Supportata
Cina (Pechino)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Cina (Ningxia)	Supportata		Supportato	Supportato		Supportato	Supportato	Supportata
Israele (Tel Aviv)			Supportata	Supportato		Supportato	Supportato	Supportato

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

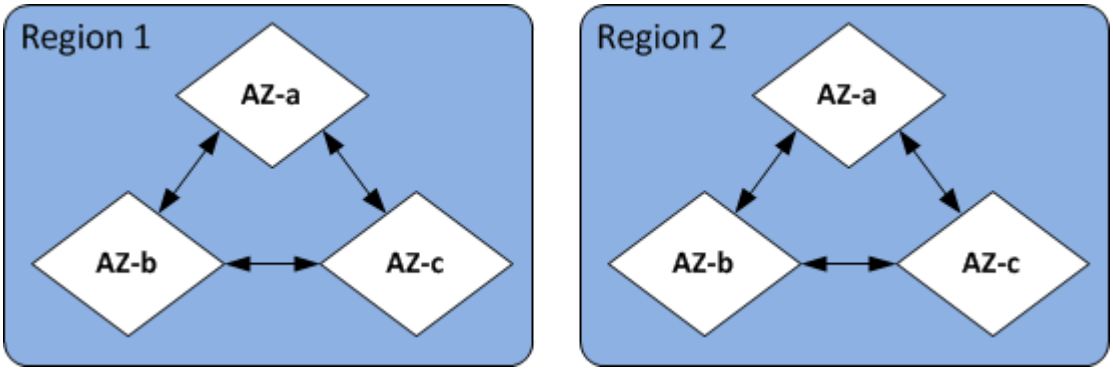
Regioni e zone di disponibilità

Regioni AWS e le zone di disponibilità definiscono le posizioni fisiche del cluster e delle istanze.

Regioni

AWS Le risorse di cloud computing sono ospitate in strutture di data center ad alta disponibilità in diverse aree del mondo (ad esempio, Nord America, Europa o Asia). La posizione di ogni data center è chiamata Regione AWS.

Regione AWS Ognuna è progettata per essere completamente isolata dall'altra Regioni AWS. All'interno di ciascuna regione sono presenti più zone di disponibilità. Avviando i nodi in diverse zone di disponibilità, puoi ottenere la massima tolleranza ai guasti possibile. Il diagramma seguente mostra una vista di alto livello del funzionamento Regioni AWS delle zone di disponibilità.



Zone di disponibilità

Ciascuna Regione AWS contiene più località distinte denominate zone di disponibilità. Ogni zona di disponibilità è progettata per essere isolata dagli errori che si verificano in altre zone di disponibilità e per offrire connettività di rete conveniente e a bassa latenza ad altre zone di disponibilità nella

stessa regione. Avviando istanze per un determinato cluster in più zone di disponibilità, è possibile proteggere le applicazioni dall'improbabile evento di errore di una zona di disponibilità.

L'architettura Amazon DocumentDB separa storage ed elaborazione. Per il livello di storage, Amazon DocumentDB replica sei copie dei dati in tre zone di disponibilità. AWS Ad esempio, se stai lanciando un cluster Amazon DocumentDB in una regione che supporta solo due zone di disponibilità, lo storage dei dati verrà replicato in sei modi su tre zone di disponibilità, ma le istanze di calcolo saranno disponibili solo in due zone di disponibilità.

La tabella seguente elenca il numero di zone di disponibilità che puoi utilizzare in una determinata Regione AWS per il provisioning delle istanze di calcolo per il tuo cluster.

Nome della regione	Regione	Zone di disponibilità (calcolo)
Stati Uniti orientali (Ohio)	`us-east-2`	3
Stati Uniti orientali (Virginia settentri onale)	`us-east-1`	6
Stati Uniti occidentali (Oregon)	`us-west-2`	4
Africa (Città del Capo)	`af-south-1`	3
Sud America (San Paolo)	`sa-east-1`	3
Asia Pacifico (Hong Kong)	`ap-east-1`	3
Asia Pacifico (Hyderabad)	`ap-south-2`	3
Asia Pacifico (Malesia)	`ap-southeast-5`	3
Asia Pacifico (Mumbai)	`ap-south-1`	3

Nome della regione	Regione	Zone di disponibilità (calcolo)
Asia Pacifico (Osaka)	ap-northeast-3	3
Asia Pacifico (Seul)	ap-northeast-2	4
Asia Pacifico (Singapore)	ap-southeast-1	3
Asia Pacifico (Sydney)	ap-southeast-2	3
Asia Pacifico (Giacarta)	ap-southeast-3	3
Asia Pacifico (Melbourne)	ap-southeast-4	3
Asia Pacifico (Thailandia)	ap-southeast-7	3
Asia Pacifico (Tokyo)	ap-northeast-1	3
Canada (Centrale)	ca-central-1	3
Canada occidentale (Calgary)	ca-west-1	3
Regione Cina (Pechino)	cn-north-1	3
Cina (Ningxia)	cn-northwest-1	3
Europa (Francoforte)	eu-central-1	3
Europa (Zurigo)	eu-central-2	3
Europa (Irlanda)	eu-west-1	3
Europa (Londra)	eu-west-2	3

Nome della regione	Regione	Zone di disponibilità (calcolo)
Europa (Milano)	`eu-south-1`	3
Europa (Parigi)	`eu-west-3`	3
Europa (Spagna)	`eu-south-2`	3
Europa (Stoccolma)	`eu-north-1`	3
Messico (centrale)	`mx-central-1`	3
Medio Oriente (Emirati Arabi Uniti)	`me-central-1`	3
Israele (Tel Aviv)	`il-central-1`	3
AWS GovCloud (US-West)	`us-gov-west-1`	3
AWS GovCloud (US-East)	`us-gov-east-1`	3

Prezzi di Amazon DocumentDB

I cluster Amazon DocumentDB vengono fatturati in base ai seguenti componenti:

- Ore di istanza (all'ora): in base alla classe dell'istanza (ad esempio). `db.r5.xlarge` I prezzi sono calcolati in base a una tariffa oraria, mentre le fatture sono calcolate al secondo e mostrano i valori in formato decimale. L'utilizzo di Amazon DocumentDB viene fatturato in incrementi di un secondo, con un minimo di 10 minuti. Per ulteriori informazioni, consulta Gestione delle classi di istanze.
- I/O richieste (per 1 milione di richieste al mese): numero totale di I/O richieste di archiviazione effettuate in un ciclo di fatturazione.
- Storage di backup (per GiB al mese): lo storage di backup è lo storage associato ai backup automatici del database e a tutte le istantanee attive del database acquisite. Estendendo il periodo di retention dei backup o creando ulteriori snapshot del database, si aumenta lo storage di backup consumato dal database. Lo spazio di archiviazione di backup viene misurato GB-months e non

viene applicato al secondo. Per ulteriori informazioni, consulta Backup e ripristino in Amazon DocumentDB.

- Trasferimento dati (per GB): trasferimento di dati in entrata e in uscita dall'istanza da o verso Internet o altro Regioni AWS.

Per informazioni dettagliate, consulta i prezzi di Amazon DocumentDB.

Prova gratuita

Puoi provare Amazon DocumentDB gratuitamente utilizzando la versione di prova gratuita di 1 mese. Per ulteriori informazioni, consulta la versione di prova gratuita sul sito dei prezzi di Amazon DocumentDB o consulta le domande frequenti sulla versione di prova gratuita di Amazon DocumentDB.

Monitoraggio

Esistono vari modi per tenere traccia delle prestazioni e dello stato di un'istanza. Puoi utilizzare il CloudWatch servizio Amazon gratuito per monitorare le prestazioni e lo stato di un'istanza. Puoi trovare i grafici delle prestazioni sulla console Amazon DocumentDB. Puoi iscriverti agli eventi di Amazon DocumentDB per ricevere una notifica quando si verificano modifiche con un'istanza, un'istantanea, un gruppo di parametri o un gruppo di sicurezza.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Monitoraggio di Amazon DocumentDB con CloudWatch
- Registrazione delle chiamate API di Amazon DocumentDB con AWS CloudTrail

Interfacce

Esistono diversi modi per interagire con Amazon DocumentDB, tra cui il e il Console di gestione AWS . AWS CLI

Console di gestione AWS

Console di gestione AWS È una semplice interfaccia utente basata sul Web. La gestione delle istanze e dei cluster dalla console non richiede alcuna programmazione. Per accedere alla console Amazon DocumentDB, accedi Console di gestione AWS e apri la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb

AWS CLI

Puoi utilizzare AWS Command Line Interface (AWS CLI) per gestire i cluster e le istanze Amazon DocumentDB. Con una configurazione minima, puoi iniziare a utilizzare tutte le funzionalità fornite dalla console Amazon DocumentDB dal tuo programma terminale preferito.

- Per installare AWS CLI, consulta Installazione dell'interfaccia a AWS riga di comando.
- Per iniziare a utilizzare AWS CLI per Amazon DocumentDB, consulta AWS Command Line Interface Reference for Amazon DocumentDB.

Driver MongoDB

Per sviluppare e scrivere applicazioni su un cluster Amazon DocumentDB, puoi anche utilizzare i driver MongoDB con Amazon DocumentDB. Per ulteriori informazioni, consulta la scheda della shell MongoDB in o. Connessione con TLS abilitato Connessione con TLS disabilitato

Fasi successive

Le sezioni precedenti hanno introdotto i componenti dell'infrastruttura di base offerti da Amazon DocumentDB. Cosa potrai fare dopo? A seconda delle circostanze, consulta uno dei seguenti argomenti per iniziare:

- Inizia a usare Amazon DocumentDB creando un cluster e un'istanza utilizzando CloudFormation Guida rapida all'utilizzo di Amazon DocumentDB CloudFormation.
- Inizia a usare Amazon DocumentDB creando un cluster e un'istanza utilizzando le istruzioni contenute nel nostro. Guida introduttiva
- Inizia a usare Amazon DocumentDB creando un cluster elastico utilizzando le istruzioni in. Inizia a usare i cluster elastici Amazon DocumentDB
- Esegui la migrazione della tua implementazione MongoDB su Amazon DocumentDB utilizzando le linee guida all'indirizzo Migrazione e aggiornamento di Amazon DocumentDB

Amazon DocumentDB: come funziona

Amazon DocumentDB (con compatibilità MongoDB) è un servizio di database completamente gestito. MongoDB-compatible Con Amazon DocumentDB, puoi eseguire lo stesso codice dell'applicazione e

utilizzare gli stessi driver e strumenti che usi con MongoDB. Amazon DocumentDB è compatibile con MongoDB 3.6, 4.0, 5.0 e 8.0.

Argomenti

- Versioni di Amazon DocumentDB
- Endpoint Amazon DocumentDB
- Supporto TLS
- Storage Amazon DocumentDB
- Replica Amazon DocumentDB
- Affidabilità di Amazon DocumentDB
- Leggi le opzioni di preferenza
- Il TTL elimina
- Risorse fatturabili

Versioni di Amazon DocumentDB

> **Note**
>
> La numerazione delle versioni secondarie si applica ad Amazon DocumentDB 5.0 e versioni successive. Amazon DocumentDB 3.6 e 4.0 utilizzano solo le versioni principali con patch del motore.

A partire da Amazon DocumentDB 5.0, la versione del motore utilizza uno schema di numerazione a quattro livelli: major.major.minor.patch.

- Versione principale: identificata dai primi due numeri della versione (ad esempio, 5.0). La versione principale rappresenta il livello di compatibilità di MongoDB. Le versioni principali introducono nuove funzionalità, miglioramenti delle prestazioni e aggiornamenti di compatibilità con MongoDB. L'aggiornamento tra le versioni principali (ad esempio, dalla 4.0 alla 5.0) richiede un aggiornamento della versione principale (MVU). Per ulteriori informazioni, consulta Aggiornamento della versione principale in-place di Amazon DocumentDB.
- Versione secondaria: identificata dal terzo numero della versione (ad esempio, in 5.0.1). 1 Le versioni secondarie includono miglioramenti come nuove funzionalità, correzioni di sicurezza e correzioni di bug all'interno della stessa versione principale. È possibile eseguire l'aggiornamento a

una nuova versione secondaria manualmente. Per ulteriori informazioni, consulta Aggiornamento della versione secondaria di Amazon DocumentDB.

- Versione della patch: le patch del motore vengono applicate all'interno di una versione secondaria e contengono correzioni critiche di sicurezza e correzioni di bug. La versione della patch viene tracciata dalla versione della patch del motore (ad esempio`3.0.17983`), che puoi verificare eseguendo. `db.runCommand({getEngineVersion: 1})` Le patch vengono applicate durante la finestra di manutenzione del cluster. Per ulteriori informazioni, consulta Manutenzione di Amazon DocumentDB.

Quando si crea un nuovo cluster, è possibile specificare qualsiasi versione del motore disponibile. Per informazioni sulle versioni disponibili e sulle date di supporto, consultaDate di supporto delle versioni del motore Amazon DocumentDB.

Quando usi Amazon DocumentDB, inizi con la creazione di un cluster. Un cluster è composto da una o più istanze database e da un volume cluster per la gestione dei dati di tali istanze. Un volume cluster Amazon DocumentDB è un volume di storage di database virtuale che si estende su più zone di disponibilità. Ogni zona di disponibilità ha una copia dei dati del cluster.

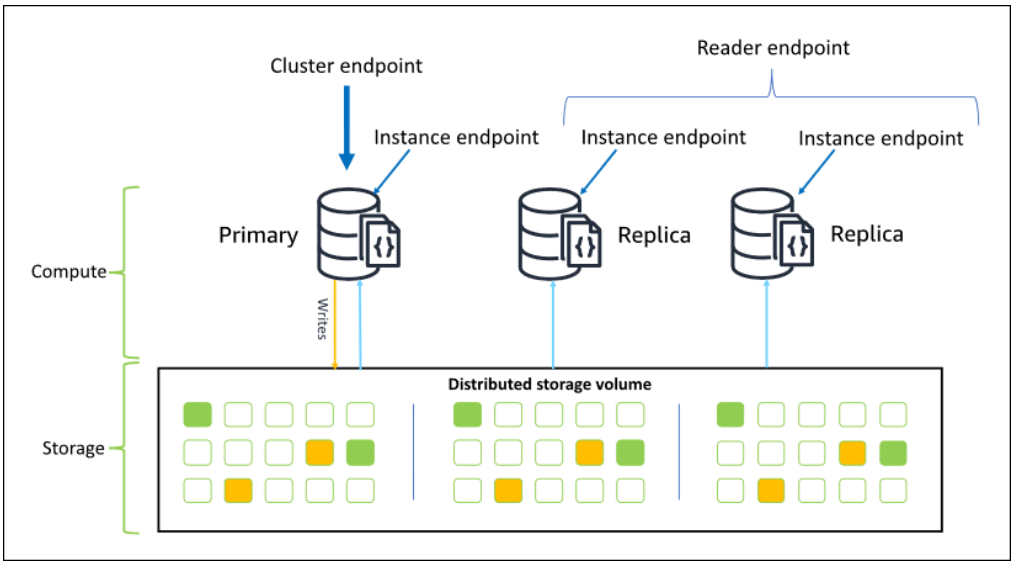
Un cluster Amazon DocumentDB è costituito da due componenti:

- Volume del cluster: utilizza un servizio di storage nativo del cloud per replicare i dati in sei modi su tre zone di disponibilità, fornendo uno storage altamente durevole e disponibile. Un cluster Amazon DocumentDB ha esattamente un volume di cluster, che può memorizzare fino a 256 TiB di dati per la versione 8.0 e successive del motore (128 TiB per le versioni precedenti del motore).
- Istanze: forniscono la potenza di elaborazione del database, scrivendo dati e leggendo i dati dal volume di storage del cluster. Un cluster Amazon DocumentDB può avere 0-16 istanze.

Le istanze sono utilizzate per uno dei due ruoli:

- Istanza primaria: supporta le operazioni di lettura e scrittura ed esegue tutte le modifiche dei dati al volume del cluster. Ogni cluster Amazon DocumentDB ha un'istanza primaria.
- Istanza di replica: supporta solo le operazioni di lettura. Un cluster Amazon DocumentDB può avere fino a 15 repliche oltre all'istanza primaria. La presenza di più repliche consente di distribuire i carichi di lavoro di lettura. Inoltre, è sufficiente collocare le repliche in zone di disponibilità separate per aumentare anche la disponibilità del cluster.

Il diagramma seguente illustra la relazione tra il volume del cluster, l'istanza primaria e le repliche in un cluster Amazon DocumentDB:



Le istanze cluster non devono appartenere alla stessa classe di istanza e possono essere sottoposte a provisioning e terminate in base alle esigenze. Questa architettura consente di aumentare la capacità di calcolo del cluster in modo indipendente dallo storage.

Quando l'applicazione scrive i dati in un'istanza primaria, questa istanza esegue una scrittura durevole nel volume cluster. Quindi replica lo stato di tale scrittura (non i dati) su ogni replica attiva. Le repliche di Amazon DocumentDB non partecipano all'elaborazione delle scritture e pertanto le repliche Amazon DocumentDB sono vantaggiose per la scalabilità in lettura. Le letture dalle repliche di Amazon DocumentDB sono infine coerenti con un ritardo minimo, in genere meno di 100 millisecondi dopo la scrittura dei dati da parte dell'istanza primaria. La funzionalità garantisce che le letture dalle repliche vengano lette nell'ordine con cui sono state scritte nell'istanza primaria. Il ritardo della replica varia a seconda della percentuale di variazione dei dati e i periodi con un'intensa attività di scrittura potrebbero aumentare il ritardo della replica. Per ulteriori informazioni, consulta la pagina relativa ai parametri `ReplicationLag` in Metriche di Amazon DocumentDB.

Endpoint Amazon DocumentDB

Amazon DocumentDB offre diverse opzioni di connessione per soddisfare un'ampia gamma di casi d'uso. Per connetterti a un'istanza in un cluster Amazon DocumentDB, devi specificare l'endpoint dell'istanza. Un endpoint è formato da un indirizzo host e da un numero di porta, separati da due punti.

Connettiti al tuo cluster utilizzando l'endpoint del cluster in modalità replica set (vedi). Connessione ad Amazon DocumentDB come set di repliche Usa questo approccio a meno che tu non debba connetterti all'endpoint del lettore o a un endpoint di istanza. Per instradare le richieste alle repliche, scegli un'impostazione di preferenza di lettura del driver che massimizza il dimensionamento della lettura nel rispetto dei requisiti di coerenza di lettura dell'applicazione. La preferenza di lettura `secondaryPreferred` consente la lettura delle repliche e libera l'istanza primaria per eseguire ulteriori operazioni.

I seguenti endpoint sono disponibili in un cluster Amazon DocumentDB.

Endpoint del cluster

L'endpoint del cluster consente la connessione all'istanza primaria corrente del cluster. L'endpoint del cluster può essere utilizzato per le operazioni di lettura e scrittura. Un cluster Amazon DocumentDB ha esattamente un endpoint del cluster.

L'endpoint del cluster fornisce il supporto per il failover per le connessioni in lettura e scrittura al cluster. Se l'istanza primaria corrente del cluster non riesce e il cluster ha almeno una replica di lettura attiva, l'endpoint del cluster reindirizza automaticamente le richieste di connessione a una nuova istanza primaria. Quando ti connetti al tuo cluster Amazon DocumentDB, utilizza l'endpoint del cluster in modalità replica set (vedi). Connessione ad Amazon DocumentDB come set di repliche

Di seguito è riportato un esempio di endpoint del cluster Amazon DocumentDB:

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

Di seguito è riportato un esempio della stringa di connessione che utilizza questo endpoint del cluster:

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

Per ulteriori informazioni sulla ricerca di un endpoint del cluster, consulta Individuazione degli endpoint di un cluster.

Endpoint di lettura

L'endpoint lettore bilancia il carico delle connessioni di sola lettura in tutte le repliche disponibili nel cluster. Un endpoint di lettura di cluster fungerà da endpoint del cluster se ci si connette tramite la `replicaSet` modalità, il che significa che nella stringa di connessione, il parametro del set di replica

è. `&replicaSet=rs0` In questo caso, sarà possibile eseguire operazioni di scrittura sul primario. Tuttavia, se ci si connette al cluster specificato`directConnection=true`, il tentativo di eseguire un'operazione di scrittura su una connessione all'endpoint del lettore genera un errore. Un cluster Amazon DocumentDB ha esattamente un endpoint di lettura.

Se il cluster contiene solo un'istanza (primaria), l'endpoint lettore si connette all'istanza primaria. Quando aggiungi un'istanza di replica al tuo cluster Amazon DocumentDB, l'endpoint di lettura apre connessioni di sola lettura alla nuova replica dopo che è attiva.

Di seguito è riportato un esempio di endpoint di lettura per un cluster Amazon DocumentDB:

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

Di seguito è riportato un esempio della stringa di connessione che utilizza un endpoint lettore:

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

L'endpoint lettore bilancia il carico delle connessioni di sola lettura, non delle richieste di lettura. Se alcune connessioni dell'endpoint lettore vengono utilizzate più di altre, le richieste di lettura potrebbero non essere equamente divise tra le istanze nel cluster. Si consiglia di distribuire le richieste collegandosi all'endpoint del cluster come set di repliche e utilizzando l'opzione preferenza di lettura secondaryPreferred.

Per ulteriori informazioni sulla ricerca di un endpoint del cluster, consulta Individuazione degli endpoint di un cluster.

Endpoint dell'istanza

Un endpoint dell'istanza si connette a un'istanza specifica all'interno del cluster. L'endpoint dell'istanza per l'istanza primaria corrente può essere utilizzato per le operazioni di lettura e scrittura. Tuttavia, il tentativo di eseguire operazioni di scrittura su un endpoint dell'istanza per una replica di lettura genera un errore. Un cluster Amazon DocumentDB ha un endpoint di istanza per istanza attiva.

Un endpoint dell'istanza fornisce il controllo diretto sulle connessioni a una specifica istanza per scenari in cui l'utilizzo di endpoint lettore o del cluster potrebbe non essere appropriato. Un caso d'uso di esempio è il provisioning per un carico di lavoro di analisi di sola lettura periodico. Puoi effettuare il provisioning di un'istanza di replica con dimensioni superiori a quelle normali, connetterti

direttamente a questa nuova istanza di grandi dimensioni con l'endpoint dell'istanza, eseguire query di analisi e terminare l'istanza. L'utilizzo dell'endpoint dell'istanza evita che il traffico delle analisi influisca negativamente sulle altre istanze del cluster.

Di seguito è riportato un esempio di endpoint di istanza per una singola istanza in un cluster Amazon DocumentDB:

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

Di seguito è riportato un esempio della stringa di connessione che utilizza questo endpoint dell'istanza:

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

> **Note**
>
> Il ruolo di un'istanza primaria o di replica può variare a causa di un evento di failover. Le applicazioni non devono mai presupporre che un determinato endpoint dell'istanza sia l'istanza primaria. Non connetterti agli endpoint delle istanze per le applicazioni di produzione. Utilizza invece l'endpoint del cluster in modalità set di repliche (vediConnessione ad Amazon DocumentDB come set di repliche). Per un controllo più avanzato sulla priorità di failover delle istanze, consulta Comprendere la tolleranza agli errori del cluster Amazon DocumentDB.

Per ulteriori informazioni sulla ricerca di un endpoint del cluster, consulta Individuazione dell'endpoint di un'istanza.

modalità Replica set

Puoi connetterti all'endpoint del tuo cluster Amazon DocumentDB in modalità set di repliche specificando il nome del set di repliche. `rs0` La connessione con la modalità per i set di repliche consente di specificare le opzioni per Read Concern, Write Concern e Read Preference. Per ulteriori informazioni, consulta Consistenza di lettura.

Di seguito è riportato un esempio della stringa di connessione connessa con la modalità per i set di repliche:

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

Quando ti connetti in modalità set di repliche, il cluster Amazon DocumentDB appare ai driver e ai client come set di repliche. Le istanze aggiunte e rimosse dal cluster Amazon DocumentDB si riflettono automaticamente nella configurazione del set di repliche.

Ogni cluster Amazon DocumentDB è costituito da un singolo set di repliche con il nome predefinito. `rs0` Il nome del set di repliche non può essere modificato.

La connessione all'endpoint del cluster con la modalità per i set di repliche è il metodo consigliato per l'uso generale.

> **Note**
>
> Tutte le istanze di un cluster Amazon DocumentDB sono in ascolto sulla stessa porta TCP per le connessioni.

Supporto TLS

Per maggiori dettagli sulla connessione ad Amazon DocumentDB tramite Transport Layer Security (TLS), consulta. Crittografia dei dati in transito

Storage Amazon DocumentDB

I dati di Amazon DocumentDB sono archiviati in un volume cluster, che è un singolo volume virtuale che utilizza unità a stato solido (SSD). Un volume del cluster è composto da sei copie dei tuoi dati, che vengono replicate automaticamente su più zone di disponibilità in un'unica. Regione AWS Questa replica contribuisce a garantire l'estrema durata dei tuoi dati e a ridurre il rischio di perdita dei dati. Consente inoltre di assicurare che il cluster non sia più disponibile durante un failover perché le copie dei dati sono già presenti in altre zone di disponibilità. Queste copie possono continuare a servire le richieste di dati alle istanze del cluster Amazon DocumentDB.

Come viene fatturato lo storage dei dati

Amazon DocumentDB aumenta automaticamente le dimensioni di un volume del cluster all'aumentare della quantità di dati. Un volume di cluster Amazon DocumentDB può raggiungere una dimensione massima di 256 TiB per la versione 8.0 e successive del motore (128 TiB per le

versioni precedenti del motore). Tuttavia, ti viene addebitato solo lo spazio utilizzato in un volume cluster Amazon DocumentDB. A partire da Amazon DocumentDB 4.0, quando i dati vengono rimossi, ad esempio eliminando una raccolta o un indice, lo spazio complessivo allocato diminuisce di una quantità comparabile. Pertanto, puoi ridurre i costi di archiviazione eliminando raccolte, indici e database che non ti servono più. In Amazon DocumentDB versione 3.6, il volume del cluster può riutilizzare lo spazio liberato quando rimuovi i dati, ma le dimensioni del volume stesso non diminuiscono mai. Di conseguenza, nella versione 3.6, potresti non riscontrare alcun cambiamento nello storage quando elimini una raccolta o un indice, anche se lo spazio liberato viene riutilizzato.

> **Note**
>
> Con Amazon DocumentDB 3.6, i costi di storage si basano sul limite massimo di storage (l'importo massimo allocato per il cluster Amazon DocumentDB in qualsiasi momento). Puoi gestire i costi evitando le pratiche ETL che creano grandi volumi di informazioni temporanee o che caricano grandi volumi di nuovi dati prima di rimuovere i vecchi dati non necessari. Se la rimozione dei dati da un cluster Amazon DocumentDB comporta una notevole quantità di spazio allocato ma inutilizzato, il ripristino del limite massimo richiede un dump logico dei dati e il ripristino in un nuovo cluster, utilizzando uno strumento come o. `mongodump` `mongorestore` La creazione e il ripristino di una snapshot non riduce lo storage allocato perché il livello fisico dello storage sottostante rimane uguale nella snapshot ripristinata.

> **Note**
>
> L'utilizzo di utilità comporta `mongodump` I/O costi aggiuntivi `mongorestore` in base alle dimensioni dei dati che vengono letti e scritti nel volume di storage.

Per informazioni sull'archiviazione dei dati e sui prezzi di Amazon DocumentDB, consulta le I/O domande frequenti sui prezzi e sui prezzi di Amazon DocumentDB (con compatibilità MongoDB). https://aws.amazon.com/documentdb/faqs/#Pricing

Replica Amazon DocumentDB

In un cluster Amazon DocumentDB, ogni istanza di replica espone un endpoint indipendente. Questi endpoint di replica forniscono l'accesso in sola lettura ai dati del volume cluster. Consentono di calibrare il carico di lavoro in lettura dei dati su più istanze replicate. Inoltre, aiutano a migliorare le prestazioni di lettura dei dati e ad aumentare la disponibilità dei dati nel cluster Amazon

DocumentDB. Le repliche di Amazon DocumentDB sono anche obiettivi di failover e vengono promosse rapidamente in caso di errore dell'istanza principale del cluster Amazon DocumentDB.

Affidabilità di Amazon DocumentDB

Amazon DocumentDB è progettato per essere affidabile, durevole e tollerante ai guasti. (Per migliorare la disponibilità, è necessario configurare il cluster Amazon DocumentDB in modo che abbia più istanze di replica in diverse zone di disponibilità). Amazon DocumentDB include diverse funzionalità automatiche che lo rendono una soluzione di database affidabile.

Riparazione automatica dello storage

Amazon DocumentDB mantiene più copie dei dati in tre zone di disponibilità, riducendo notevolmente la possibilità di perdita dei dati a causa di un errore di archiviazione. Amazon DocumentDB rileva automaticamente i guasti nel volume del cluster. Quando un segmento di un volume del cluster si guasta, Amazon DocumentDB ripara immediatamente il segmento. Utilizza i dati degli altri volumi che costituiscono il volume cluster per garantire che i dati del segmento riparato siano aggiornati. Di conseguenza, Amazon DocumentDB evita la perdita di dati e riduce la necessità di eseguire un ripristino point-in-time per il ripristino in caso di errore di un'istanza.

Riscaldamento di sopravvivenza della cache

Amazon DocumentDB gestisce la cache delle pagine in un processo separato dal database in modo che la cache delle pagine possa sopravvivere indipendentemente dal database. Nella remota eventualità di un errore del database, la cache di pagina rimane in memoria. In questo modo il pool di buffer viene riscaldato con lo stato più aggiornato quando il database viene riavviato.

Ripristino in caso di arresto

Amazon DocumentDB è progettato per il ripristino da un arresto anomalo quasi istantaneamente e per continuare a fornire i dati delle applicazioni. Amazon DocumentDB esegue il crash recovery in modo asincrono su thread paralleli in modo che il database sia aperto e disponibile quasi immediatamente dopo un arresto anomalo.

Governance delle risorse

Amazon DocumentDB protegge le risorse per processi critici come i controlli sullo stato di salute. Quando un'istanza presenta una pressione di memoria elevata, Amazon DocumentDB limita le richieste. Alcune operazioni potrebbero rimanere in coda fino a quando la pressione della memoria non diminuisce. Se la pressione della memoria continua, le operazioni in coda potrebbero scadere.

Monitora la limitazione della memoria insufficiente con le seguenti CloudWatch metriche:,,,. `LowMemThrottleQueueDepth` `LowMemThrottleMaxQueueDepth` `LowMemNumOperationsThrottled` `LowMemNumOperationsTimedOut` Per ulteriori informazioni, consulta Monitoraggio di Amazon DocumentDB con CloudWatch.

Se riscontri una pressione di memoria sostenuta, espandi l'istanza per fornire memoria aggiuntiva per il carico di lavoro.

Leggi le opzioni di preferenza

Amazon DocumentDB utilizza un servizio di storage condiviso nativo per il cloud che replica i dati sei volte in tre zone di disponibilità per fornire livelli elevati di durabilità. Amazon DocumentDB non si basa sulla replica dei dati su più istanze per ottenere la durabilità. I dati del cluster sono sempre durevoli, sia che contengano una singola istanza che 15.

Argomenti

- Durata della scrittura
- Isolamento in lettura
- Consistenza di lettura
- Elevata disponibilità
- Il ridimensionamento delle letture

Durata della scrittura

Amazon DocumentDB utilizza un sistema di storage unico, distribuito, tollerante ai guasti e con risoluzione automatica. Questo sistema replica sei copie (V=6) dei dati in tre AWS zone di disponibilità per fornire disponibilità e durata elevate. Durante la scrittura dei dati, Amazon DocumentDB garantisce che tutte le scritture siano registrate in modo duraturo sulla maggior parte dei nodi prima di riconoscere la scrittura al client. Se utilizzi un set di repliche MongoDB a tre nodi, l'utilizzo di un criterio di scrittura di `{w:3, j:true}` produrrebbe la migliore configurazione possibile rispetto ad Amazon DocumentDB.

Le scritture su un cluster Amazon DocumentDB devono essere elaborate dall'istanza di writer del cluster. Il tentativo di scrivere su un lettore genera un errore. Una scrittura riconosciuta da un'istanza primaria di Amazon DocumentDB è durevole e non può essere ripristinata. Amazon DocumentDB è altamente durevole per impostazione predefinita e non supporta un'opzione di scrittura non durevole.

Non puoi modificare il livello di durabilità (ovvero Write Concern). Amazon DocumentDB ignora w=anything ed è effettivamente w: 3 e j: true. Non puoi ridurlo.

Poiché lo storage e l'elaborazione sono separati nell'architettura Amazon DocumentDB, un cluster con una singola istanza è estremamente durevole. La durabilità è gestita a livello di storage. Di conseguenza, un cluster Amazon DocumentDB con una singola istanza e uno con tre istanze raggiungono lo stesso livello di durabilità. Puoi configurare il cluster per il tuo caso d'uso specifico senza sacrificare la durabilità elevata per i dati.

Le scritture su un cluster Amazon DocumentDB sono atomiche all'interno di un singolo documento.

Amazon DocumentDB non supporta l'`wtimeout`opzione e non restituirà un errore se viene specificato un valore. È garantito che le scritture sull'istanza Amazon DocumentDB primaria non si bloccano a tempo indeterminato.

Isolamento in lettura

Le letture da un'istanza Amazon DocumentDB restituiscono solo dati durevoli prima dell'inizio della query. Le letture non restituiscono mai i dati modificati dopo aver avviato l'esecuzione delle query e in nessun caso sono consentite le letture modificate.

Consistenza di lettura

I dati letti da un cluster Amazon DocumentDB sono durevoli e non verranno ripristinati. Puoi modificare la coerenza di lettura per le letture di Amazon DocumentDB specificando la preferenza di lettura per la richiesta o la connessione. Amazon DocumentDB non supporta un'opzione di lettura non duratura.

Le letture dall'istanza principale di un cluster Amazon DocumentDB sono fortemente coerenti in condizioni operative normali e hanno una coerenza lettura dopo scrittura. Se si verifica un failover tra la scrittura e la lettura successiva, il sistema è in grado di restituire per breve tempo una lettura non particolarmente coerente. Tutte le letture da una replica di lettura garantiscono coerenza finale e restituiscono i dati nello stesso ordine, spesso con un ritardo di replica inferiore ai 100 ms.

Preferenze di lettura di Amazon DocumentDB

Amazon DocumentDB supporta l'impostazione di un'opzione di preferenza di lettura solo durante la lettura dei dati dall'endpoint del cluster in modalità set di repliche. L'impostazione di un'opzione di preferenza di lettura influisce sul modo in cui il client o il driver MongoDB indirizza le richieste di lettura alle istanze del cluster Amazon DocumentDB. Puoi impostare le opzioni per le preferenze di lettura per una query specifica oppure come opzione generale nel tuo driver MongoDB. Consulta la

documentazione del driver o del client per istruzioni su come impostare un'opzione per le preferenze di lettura.

Se il client o il driver non si connette a un endpoint del cluster Amazon DocumentDB in modalità set di repliche, il risultato della specifica di una preferenza di lettura non è definito.

Amazon DocumentDB non supporta l'impostazione di set di tag come preferenza di lettura.

Opzioni di preferenza di lettura supportate

- **`primary`**: la specifica di una preferenza di `primary` lettura aiuta a garantire che tutte le letture vengano indirizzate all'istanza primaria del cluster. Se l'istanza primaria non è disponibile, l'operazione di lettura non riesce. Una preferenza di lettura `primary` genera la consistenza lettura dopo scrittura ed è opportuna per i casi d'uso che assegnano priorità alla consistenza lettura dopo scrittura rispetto all'alta disponibilità e al dimensionamento della lettura.

  L'esempio seguente specifica una preferenza di lettura `primary`:

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

- **`primaryPreferred`**—Se si specifica una preferenza di `primaryPreferred` lettura, le letture vengono indirizzate all'istanza primaria in condizioni di normale funzionamento. Se si verifica un failover dell'istanza primaria, il client indirizza le richieste a una replica. Una preferenza di lettura `primaryPreferred` genera una consistenza lettura dopo scrittura durante il normale funzionamento e letture consistenti finali durante un evento di failover. Una preferenza di lettura `primaryPreferred` è opportuna per i casi d'uso che assegnano priorità alla consistenza lettura dopo scrittura rispetto al dimensionamento della lettura, ma a cui è comunque necessaria l'alta disponibilità.

  L'esempio seguente specifica una preferenza di lettura `primaryPreferred`:

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

- **`secondary`**—La specificazione di una preferenza di `secondary` lettura garantisce che le letture vengano indirizzate solo a una replica, mai all'istanza primaria. Se in un cluster non sono presenti istanze di replica, la richiesta di lettura non riesce. Una preferenza di lettura `secondary` genera

letture consistenti finali ed è opportuna per i casi d'uso che assegnano priorità al throughput di scrittura dell'istanza primaria rispetto all'alta disponibilità e alla consistenza lettura dopo scrittura.

L'esempio seguente specifica una preferenza di lettura `secondary`:

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

- **`secondaryPreferred`**—La specificazione di una preferenza di `secondaryPreferred` lettura garantisce che le letture vengano indirizzate a una replica di lettura quando una o più repliche sono attive. Se in un cluster non sono presenti istanze di replica attive, la richiesta di lettura viene instradata all'istanza primaria. Una preferenza di lettura `secondaryPreferred` genera letture consistenti finali quando la lettura viene gestita da una replica di lettura. Genera una consistenza lettura dopo scrittura quando la lettura viene gestita dall'istanza primaria (salvo eventi di failover). Una preferenza di lettura `secondaryPreferred` è opportuna per i casi d'uso che assegnano priorità alla consistenza lettura dopo scrittura rispetto all'alta disponibilità e al dimensionamento della lettura.

  L'esempio seguente specifica una preferenza di lettura `secondaryPreferred`:

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

- **`nearest`**—La specificazione di una preferenza di `nearest` lettura indirizza le letture in base esclusivamente alla latenza misurata tra il client e tutte le istanze nel cluster Amazon DocumentDB. Una preferenza di lettura `nearest` genera letture consistenti finali quando la lettura viene gestita da una replica di lettura. Genera una consistenza lettura dopo scrittura quando la lettura viene gestita dall'istanza primaria (salvo eventi di failover). Una preferenza di lettura `nearest` è opportuna per i casi d'uso che assegnano priorità al raggiungimento della minor latenza di lettura possibile e all'alta disponibilità rispetto alla consistenza lettura dopo scrittura e al dimensionamento della lettura.

  L'esempio seguente specifica una preferenza di lettura `nearest`:

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

Elevata disponibilità

Amazon DocumentDB supporta configurazioni di cluster ad alta disponibilità utilizzando le repliche come destinazioni di failover per l'istanza primaria. Se l'istanza primaria fallisce, una replica di Amazon DocumentDB viene promossa come nuova primaria, con una breve interruzione durante la quale le richieste di lettura e scrittura effettuate all'istanza primaria falliscono con un'eccezione.

Se il tuo cluster Amazon DocumentDB non include alcuna replica, l'istanza primaria viene ricreata durante un errore. Poiché la promozione di una replica Amazon DocumentDB è molto più veloce rispetto alla ricreazione dell'istanza primaria, crea una o più repliche Amazon DocumentDB come destinazioni di failover.

Le repliche che sono pensate per l'utilizzo come destinazioni di failover devono essere della stessa classe di istanza dell'istanza primaria. Devono essere assegnate in diverse zone di disponibilità dal database primario. Puoi specificare le repliche preferite come destinazioni di failover. Per le migliori pratiche sulla configurazione di Amazon DocumentDB per l'alta disponibilità, consulta. Comprendere la tolleranza agli errori del cluster Amazon DocumentDB

Il ridimensionamento delle letture

Le repliche di Amazon DocumentDB sono ideali per il ridimensionamento in lettura. Sono dedicate completamente alle operazioni di lettura nel volume cluster, ossia le repliche non elaborano le scritture. Le repliche dei dati si verificano all'interno del volume cluster e non tra le istanze. Quindi le risorse di ogni replica sono dedicate all'elaborazione delle query, non a replicare e scrivere dati.

Se l'applicazione richiede più capacità di lettura, puoi aggiungere rapidamente una replica al cluster, in genere in meno di 10 minuti. Se i tuoi requisiti per la capacità di lettura diminuiscono, puoi rimuovere le repliche non necessarie. Con le repliche di Amazon DocumentDB, paghi solo per la capacità di lettura di cui hai bisogno.

Amazon DocumentDB supporta il ridimensionamento della lettura sul lato client tramite l'uso delle opzioni di preferenza di lettura. Per ulteriori informazioni, consulta Preferenze di lettura di Amazon DocumentDB.

Il TTL elimina

Le eliminazioni da un'area di indice TTL ottenute tramite un processo in background si basano sul miglior tentativo e non sono garantite all'interno di un determinato periodo di tempo specifico. Fattori come le dimensioni dell'istanza, l'utilizzo di risorse dell'istanza, le dimensioni documento e il throughput complessivo possono influenzare la tempistica di un'eliminazione TTL.

Quando il monitor TTL elimina i documenti, ogni eliminazione comporta costi di IO incrementando l'importo in fattura. Se la velocità effettiva e i tassi di eliminazione del TTL aumentano, dovresti aspettarti un aumento della fattura a causa del maggiore utilizzo degli I/O.

Quando si crea un indice TTL su una raccolta esistente, è necessario eliminare tutti i documenti scaduti prima di creare l'indice. L'attuale implementazione TTL è ottimizzata per l'eliminazione di una piccola parte dei documenti della raccolta, il che è tipico se il TTL è stato abilitato nella raccolta sin dall'inizio e può comportare IOPS più elevati del necessario se è necessario eliminare un numero elevato di documenti contemporaneamente.

Se non desideri creare un indice TTL per eliminare i documenti, puoi invece segmentare i documenti in raccolte in base al tempo e semplicemente eliminare quelle raccolte quando i documenti non sono più necessari. Ad esempio: puoi creare una raccolta a settimana e eliminarla senza incorrere in costi di I/O. Questo può essere molto più conveniente rispetto all'utilizzo di un indice TTL.

Risorse fatturabili

Identificazione delle risorse Amazon DocumentDB fatturabili

Essendo un servizio di database completamente gestito, Amazon DocumentDB addebita le spese per le istanze, lo storage I/Os, i backup e il trasferimento dei dati. Per ulteriori informazioni, consulta i prezzi di Amazon DocumentDB (con compatibilità MongoDB).

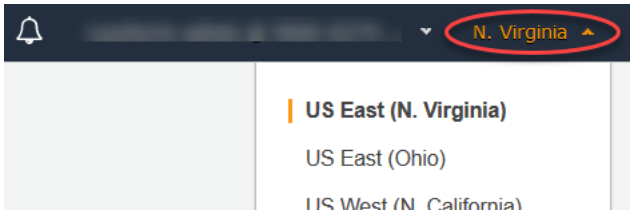
Per scoprire le risorse fatturabili nel tuo account e potenzialmente eliminarle, puoi utilizzare il comando o. Console di gestione AWS AWS CLI

Utilizzo di Console di gestione AWS

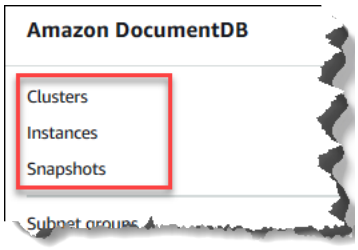
Usando il Console di gestione AWS, puoi scoprire i cluster, le istanze e gli snapshot di Amazon DocumentDB di cui hai effettuato il provisioning per un determinato periodo. Regione AWS

Per individuare cluster, istanze e snapshot

1. Accedi a e apri la console Amazon Console di gestione AWS DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Per scoprire le risorse fatturabili in una regione diversa dalla tua regione predefinita, nell'angolo in alto a destra dello schermo, scegli Regione AWS quella che desideri cercare.



3. Nel riquadro di navigazione, scegliere il tipo di risorse fatturabili di interesse: Clusters (Cluster), Instances (Istanze) o Snapshots (Snapshot).



4. Tutti i cluster, le istanze o le snapshot di cui è stato eseguito il provisioning per la regione sono elencati nel riquadro a destra. Verrà addebitato il costo per i cluster, le istanze e le snapshot.

Utilizzo di AWS CLI

Usando il AWS CLI, puoi scoprire i cluster, le istanze e le istantanee di Amazon DocumentDB di cui hai effettuato il provisioning per un determinato periodo. Regione AWS

Per individuare cluster e istanze

Il codice seguente elenca tutti i cluster e le istanze per la regione specificata. Per eseguire la ricerca di cluster e istanze nella regione predefinita, è possibile omettere il parametro `--region`.

Example

Per Linux, macOS o Unix:

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

Per Windows:

```
aws docdb describe-db-clusters ^
```

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

L'aspetto dell'output di questa operazione è simile al seguente.

```
"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",
```

Per individuare le snapshot

Il codice seguente elenca tutte le snapshot per la regione specificata. Per eseguire la ricerca di snapshot nella regione predefinita, è possibile omettere il parametro `--region`.

Per Linux, macOS o Unix:

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

Per Windows:

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

L'aspetto dell'output di questa operazione è simile al seguente.

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

```
        "manual"
    ]
]
```

È necessario eliminare solo le snapshot di tipo `manual`. Le snapshot di tipo `Automated` vengono eliminate quando si elimina il cluster.

Eliminazione di risorse fatturabili indesiderate

Per eliminare un cluster, è necessario eliminare prima tutte le istanze che esso contiene.

- Per eliminare le istanze, consultare Eliminazione di un'istanza Amazon DocumentDB.

> **Important**
>
> Anche eliminando le istanze in un cluster, lo storage e l'utilizzo di backup di utilizzo associati a tale cluster vengono comunque fatturati. Per interrompere ogni addebito, è necessario eliminare anche il cluster e le snapshot manuali.

- Per eliminare i cluster, consultare Eliminazione di un cluster Amazon DocumentDB.
- Per eliminare le snapshot manuali, consultare Eliminazione di un'istantanea del cluster.

Cos'è un database di documenti?

Alcuni sviluppatori non pensano al proprio modello di dati in termini di righe e colonne normalizzate. Tipicamente, nel livello dell'applicazione, i dati sono rappresentati come un documento JSON perché è più intuitivo per gli sviluppatori pensare al proprio modello di dati come a un documento.

La popolarità dei database di documenti è aumentata perché consentono di rendere permanenti i dati in un database utilizzando lo stesso formato del modello di documento utilizzato nel codice dell'applicazione. I database di documenti forniscono API potenti e intuitive per uno sviluppo flessibile e agile.

Argomenti

- Casi d'uso del database di documenti
- Comprensione dei documenti
- Utilizzo dei documenti

Casi d'uso del database di documenti

Il tuo caso d'uso riguarda situazioni in cui hai bisogno di un database di documenti o di altri tipi di database per gestire i dati. I database di documenti sono utili per i carichi di lavoro che richiedono uno schema flessibile per uno sviluppo rapido e iterativo. Di seguito sono elencati alcuni esempi di casi d'uso per i quali i database di documenti possono offrire notevoli vantaggi:

Argomenti

- Profili utente
- Real-time big data
- Gestione dei contenuti

Profili utente

Poiché i database di documenti si basano su uno schema flessibile, possono archiviare documenti con diversi attributi e valori dei dati. I database di documenti rappresentano sono una soluzione pratica per i profili online in cui più utenti forniscono tipi di informazioni diversi. Con un database di documenti puoi archiviare ogni profilo utente in modo efficiente archiviando solo gli attributi specifici per il singolo utente.

Supponiamo che un utente decida di aggiungere o rimuovere delle informazioni dal profilo. In questo caso, il documento può essere facilmente sostituito con una versione aggiornata che contiene tutti i dati e gli attributi aggiunti di recente o che omette attributi e dati rimossi di recente. I database di documenti consentono di gestire più facilmente questo livello di individualità e fluidità.

Real-time big data

Storicamente, la capacità di estrarre informazioni dai dati operativi era ostacolata dal fatto che i database operativi e i database analitici venivano mantenuti in ambienti diversi, operativi e rispettivamente. business/reporting La possibilità di estrarre informazioni operative in tempo reale è fondamentale in un ambiente aziendale altamente competitivo. Utilizzando i database di documenti, un'azienda può archiviare e gestire i dati operativi da qualsiasi origine e, allo stesso tempo, inviare i dati al motore BI preferito per analizzarli. La presenza di due ambienti non è obbligatoria.

Gestione dei contenuti

Per gestire in modo efficiente i contenuti, devi poterli raccogliere e aggregare da diverse origini e quindi inviarli al cliente. Grazie al loro schema flessibile, i database di documenti sono perfetti

per raccogliere e archiviare qualsiasi tipo di dati. Puoi utilizzarli per creare e integrare nuovi tipi di contenuti, inclusi quelli generati dagli utenti, come immagini, commenti e video.

Comprensione dei documenti

I database di documenti vengono utilizzati per archiviare dati semistrutturati come documento, anziché normalizzare i dati su più tabelle, ognuna con una struttura unica e fissa, come in un database relazionale. I documenti archiviati in un database di documenti utilizzano coppie chiave-valore nidificate per fornire la struttura o lo schema del documento. Tuttavia, nello stesso database di documenti possono essere archiviati diversi tipi di documenti, rispettando così il requisito che prevede l'elaborazione di dati simili in formati diversi. Ad esempio, poiché ogni documento è autodescrittivo, i JSON-encoded documenti di un negozio online descritti nell'argomento Documenti di esempio in un database di documenti possono essere archiviati nello stesso database di documenti.

Argomenti

- SQL e terminologia non relazionale
- Documenti semplici
- Documenti incorporati
- Documenti di esempio in un database di documenti
- Comprendere la normalizzazione in un database di documenti

SQL e terminologia non relazionale

Nella tabella seguente viene confrontata la terminologia utilizzata dai database di documenti (MongoDB) con quella utilizzata dai database SQL.

SQL	MongoDB
Tabella	Raccolta
Riga	Documento
Colonna	Campo
Chiave primaria	ObjectId
Indice	Indice

SQL	MongoDB
Vista	Vista
Oggetto o tabella annidata	Documento incorporato
Array	Array

Documenti semplici

Tutti i documenti in un database di documenti sono autodescrittivi. Questa documentazione utilizza documenti JSON-like formattati, sebbene sia possibile utilizzare altri metodi di codifica.

Un documento semplice ha uno o più campi, tutti allo stesso livello all'interno del documento. In questo esempio i campi `SSN`, `LName`, `FName`, `DOB`, `Street`, `City`, `State-Province`, `PostalCode` e `Country` sono tutti di pari livello all'interno del documento.

```
{
   "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"
}
```

Quando le informazioni vengono organizzate in un documento semplice, ogni campo viene gestito individualmente. Per recuperare l'indirizzo di una persona, devi recuperare `Street`, `City`, `State-Province`, `PostalCode` e `Country` come singoli elementi dati.

Documenti incorporati

Un documento complesso organizza i propri dati creando documenti incorporati nel documento stesso. I documenti incorporati consentono di gestire i dati in gruppi e come singoli elementi, in base alla soluzione più efficace in un determinato caso. Utilizzando l'esempio precedente, puoi incorporare un documento `Address` nel documento principale. In questo modo si otterrà la struttura del documento seguente:

```
{
   "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"
   }
}
```

È ora possibile accedere ai dati del documento come campi singoli (`"SSN":`), come documento incorporato (`"Address":`) o come membro di un documento incorporato (`"Address": {"Street":}`).

Documenti di esempio in un database di documenti

Come indicato in precedenza, ogni documento in un database di documenti è autodescrittivo, quindi la struttura dei documenti all'interno di un database di documenti può essere diversa. I due documenti seguenti, uno relativo a un manuale e l'altro a un periodico, sono strutturalmente differenti. Tuttavia, entrambi possono essere inclusi nello stesso database di documenti.

Di seguito è riportato un documento di esempio per un libro:

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

Di seguito è riportato un documento di esempio per un periodico con due articoli:

```
{
    "_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"
}
```

Confronta la struttura di questi due documenti. Con un database relazionale, devi avere tabelle separate per "periodico" e "libri" oppure una singola tabella con campi inutilizzati, ad esempio "Pubblicazione", "Numero", "Articoli" e "MI", come valori `null`. Poiché i database di documenti sono semistrutturati e ogni documento definisce autonomamente la propria struttura, questi due documenti possono coesistere nello stesso database di documenti senza alcun campo `null`. I database di documenti sono uno strumento efficace per gestire i dati di tipo sparse.

L'utilizzo di un database di documenti consente uno sviluppo rapido e iterativo. Il motivo è che puoi modificare la struttura dei dati di un documento in modo dinamico, senza dover modificare lo schema per l'intera raccolta. I database di documenti sono particolarmente adatti per lo sviluppo agile e gli ambienti che evolvono dinamicamente.

Comprendere la normalizzazione in un database di documenti

I database di documenti non sono normalizzati. I dati nel documento possono essere ripetuti in un altro documento. Inoltre, possono esistere delle discrepanze nei dati dei documenti. Ad esempio, considera uno scenario in cui puoi effettuare acquisti in uno store online e tutti i dettagli dei tuoi acquisti vengono archiviati in un singolo documento. Il documento può essere simile al documenti JSON seguente:

```
{
    "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"
}
```

Tutte le informazioni vengono archiviate sotto forma di documento in una raccolta di transazioni. Successivamente, ti accorgi di aver dimenticato di acquistare un articolo. Accedi di nuovo allo stesso store ed effettui un altro acquisto, che a sua volta viene archiviato come documento nella raccolta di transazioni.

```
{
    "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"
}
```

Nota la ridondanza tra questi due documenti: il tuo nome e l'ID acquirente (e, se hai usato la stessa carta di credito, i dati della carta di credito). Tuttavia, la procedura non presenta problemi perché lo storage è economico e ogni documento registra completamente una singola transazione che può essere recuperata rapidamente con una semplice query chiave-valore che non richiede join.

C'è anche un'apparente discrepanza tra i due documenti: i dati della tua carta di credito. Si tratta di una discrepanza solo apparente perché è probabile che hai semplicemente utilizzato due carte diverse per gli acquisti. Ogni documento contiene tutti i dati corretti per la transazione documentata.

Utilizzo dei documenti

Essendo un database di documenti, Amazon DocumentDB semplifica l'archiviazione, l'interrogazione e l'indicizzazione dei dati JSON. In Amazon DocumentDB, una raccolta è analoga a una tabella in un database relazionale, tranne per il fatto che non esiste un unico schema applicato a tutti i documenti. Le raccolte consentono di raggruppare documenti simili mantenendoli comunque nello stesso database, senza richiedere che siano identici a livello di struttura.

Utilizzando i documenti di esempio delle sezioni precedenti, è probabile che tu disponga di raccolte per `reading_material` e `office_supplies`. È responsabilità del software stabilire a quale raccolta appartenga un documento.

Gli esempi seguenti utilizzano l'API MongoDB per mostrare come aggiungere, eseguire query, aggiornare ed eliminare i documenti.

Argomenti

- Aggiungere documenti
- Interrogazione di documenti
- Aggiornamento dei documenti
- Eliminazione di documenti

Aggiungere documenti

In Amazon DocumentDB, un database viene creato quando si aggiunge per la prima volta un documento a una raccolta. In questo esempio, viene creata una raccolta denominata `example` nel database `test`, che è il database predefinito quando ci si connette a un cluster. Poiché la raccolta viene creata implicitamente quando viene inserito il primo documento, non viene effettuato alcun controllo degli errori sul nome della raccolta. Di conseguenza, un errore di battitura nel nome della raccolta, ad esempio `eexample` anziché `example`, creerà e aggiungerà il documento alla raccolta `eexample` anziché la raccolta desiderata. Il controllo degli errori deve essere gestito dall'applicazione.

Gli esempi seguenti utilizzano l'API MongoDB per aggiungere documenti.

Argomenti

- Aggiungere un singolo documento
- Aggiungere più documenti

Aggiungere un singolo documento

Per aggiungere un singolo documento a una raccolta, utilizza l'operazione `insertOne( {} )` con il documento da aggiungere alla raccolta.

```
db.example.insertOne(
    {
```

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

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

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

Aggiungere più documenti

Per aggiungere più documenti a una raccolta, utilizza l'operazione `insertMany( [{},...,{}] )` con un elenco di documenti da aggiungere alla raccolta. Anche se i documenti in questo specifico elenco hanno schemi diversi, ciascuno può essere aggiunto alla stessa raccolta.

```
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
            }
        }
    ]
)
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

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

Interrogazione di documenti

A volte, potrebbe essere necessario cercare nell'inventario dello store online, in modo che i clienti possano vedere e acquistare ciò che vendi. Eseguire query su una raccolta è relativamente semplice, sia per le ricerche su tutti i documenti della raccolta che per quelle solo sui documenti che soddisfano un determinato criterio.

Per eseguire una query per i documenti, utilizza l'operazione `find()`. Il comando `find()` ha un parametro per documenti singoli che definisce i criteri da utilizzare nella scelta dei documenti da restituire. L'output di `find()` è un documento formattato su una sola riga di testo senza interruzioni di riga. Per formattare il documento di output per facilitare la lettura, utilizza `find().pretty()`. Tutti gli esempi di questo argomento utilizzano `.pretty()` per formattare l'output.

Usa i quattro documenti che hai inserito nella `example` raccolta nei due esercizi precedenti: `insertOne()` e. `insertMany()`

Argomenti

- Recupero di tutti i documenti di una raccolta
- Recupero di documenti che corrispondono a un valore di campo

- Recupero di documenti che corrispondono a un documento incorporato
- Recupero di documenti che corrispondono a un valore di campo in un documento incorporato
- Recupero di documenti che corrispondono a un array
- Recupero di documenti che corrispondono a un valore in una matrice
- Recupero di documenti tramite operatori

Recupero di tutti i documenti di una raccolta

Per recuperare tutti i documenti nella raccolta, usa l'operazione `find()` con un documento di query vuoto.

La query seguente restituisce tutti i documenti della raccolta `example`.

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

Recupero di documenti che corrispondono a un valore di campo

Per recuperare tutti i documenti che corrispondono a un campo e a un valore, usa l'operazione `find()` con un documento di query che identifica i campi e i valori per la corrispondenza.

Usando i documenti precedenti, questa query restituisce tutti i documenti in cui il campo "Item" è uguale a "Pen".

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

Recupero di documenti che corrispondono a un documento incorporato

Per trovare tutti i documenti che corrispondono a un documento incorporato, utilizza l'operazione `find()` con un documento di query che specifica il nome del documento incorporato e tutti i campi e i valori per quel documento incorporato.

Quando si esegue il confronto di un documento incorporato, il documento incorporato del documento deve avere lo stesso nome che ha nella query. Inoltre, i campi e i valori nel documento incorporato devono corrispondere alla query.

La seguente query restituisce solo il documento "Poster Paint". Questo perché "Pen" ha valori diversi per "`OnHand`" e "`MinOnHand`" e "Spray Paint" ha un ulteriore campo (`OrderQnty`) rispetto al documento di query.

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

Recupero di documenti che corrispondono a un valore di campo in un documento incorporato

Per trovare tutti i documenti che corrispondono a un documento incorporato, utilizza l'operazione `find()` con un documento di query che specifica il nome del documento incorporato e tutti i campi e i valori per quel documento incorporato.

Considerati i documenti precedenti, la seguente query utilizza la "dot notation" (notazione col punto) per specificare il documento incorporato e i campi di interesse. Vengono restituiti tutti i documenti che corrispondono a questi campi, indipendentemente da quali altri campi possono essere presenti nel documento incorporato. La query restituisce "Poster Paint" e "Spray Paint" perché entrambi corrispondono ai campi e ai valori specificati.

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

Recupero di documenti che corrispondono a un array

Per trovare tutti i documenti che corrispondono a una matrice, utilizzare l'operazione `find()` con il nome della matrice richiesta e tutti i valori in quella matrice. La query restituisce tutti i documenti che hanno una matrice con quel nome in cui i valori della matrice sono identici e nello stesso ordine rispetto alla query.

La seguente query restituisce solo "Pen" perché" Poster Paint" ha un ulteriore colore (White) mentre "Spray Paint" ha i colori in un ordine diverso.

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

Recupero di documenti che corrispondono a un valore in una matrice

Per trovare tutti i documenti che hanno un valore specifico di matrice, utilizza l'operazione `find()` con il nome della matrice e il valore richiesto.

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

L'operazione precedente restituisce tutti e tre i documenti, in quanto ciascuno di essi dispone di una matrice denominata `Colors` e del valore "`Red`" all'interno della matrice. Se si specifica il valore "`White`", la query restituisce solo "Poster Paint".

Recupero di documenti tramite operatori

La seguente query restituisce tutti i documenti in cui il valore "`Inventory.OnHand`" è inferiore a 50.

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

Per un elenco degli operatori di query supportati, consulta Operatori di interrogazione e proiezione.

Aggiornamento dei documenti

In genere, i documenti non sono statici e vengono aggiornati come parte dei flussi di lavoro dell'applicazione. Gli esempi seguenti mostrano alcune opzioni di aggiornamento dei documenti.

Per aggiornare un documento esistente, utilizza l'operazione `update()`. L'operazione `update()` presenta due parametri del documento. Il primo documento identifica il documento o i documenti da aggiornare. Il secondo documento specifica gli aggiornamenti da eseguire.

Quando si aggiorna un campo esistente, indipendentemente dal fatto che si tratti di un campo semplice, di un array o di un documento incorporato, si specifica il nome del campo e i relativi valori. Al termine dell'operazione è come se il campo nel documento precedente fosse stato sostituito dal campo e dai valori nuovi.

Argomenti

- Aggiornamento dei valori di un campo esistente
- Aggiungere un nuovo campo
- Sostituzione di un documento incorporato
- Inserimento di nuovi campi in un documento incorporato
- Rimozione di un campo da un documento
- Rimozione di un campo da più documenti

Aggiornamento dei valori di un campo esistente

Utilizza i seguenti quattro documenti aggiunti in precedenza per le seguenti operazioni di aggiornamento.

```
{
    "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
    }
}
```

Per aggiornare un campo semplice

Per aggiornare un campo semplice, utilizza `update()` con `$set` per specificare il nome del campo e il nuovo valore. L'esempio seguente modifica l'elemento `Item` da "Pen" a "Gel Pen".

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

L'aspetto dei risultati di questa operazione è simile al seguente.

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

Per aggiornare una matrice

L'esempio seguente sostituisce la matrice esistente di colori con una nuova matrice che comprende `Orange` e rimuove `White` dall'elenco dei colori. Il nuovo elenco di colori è nell'ordine specificato nell'operazione `update()`.

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

L'aspetto dei risultati di questa operazione è simile al seguente.

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

Aggiungere un nuovo campo

Per modificare un documento aggiungendo uno o più nuovi campi, utilizza l'operazione `update()` con un documento di query che identifica il documento in cui inserire i dati e i nuovi campi e valori da inserire utilizzando l'operatore `$set`.

L'esempio seguente aggiunge il campo `UnitPrice` con il valore `3.99` al documento Spray Paints. Si noti che il valore `3.99` è numerico e non una stringa.

```
db.example.update(
```

```
    { "Item": "Spray Paint" },
    { $set: { "UnitPrice": 3.99 } }
)
```

I risultati di questa operazione sono simili ai seguenti (formato JSON).

```
{
    "Item": "Spray Paint",
    "Colors": ["Black","Red","Green","Blue"],
    "Inventory": {
        "OnHand": 47,
        "MinOnHand": 50,
        "OrderQnty": 36
    },
    "UnitPrice": 3.99
}
```

Sostituzione di un documento incorporato

Per modificare un documento sostituendo un documento incorporato, utilizza l'operazione `update()` con i documenti che identificano il documento incorporato e i nuovi campi e valori utilizzando l'operatore `$set`.

Considerato il seguente documento.

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

Per sostituire un documento incorporato

L'esempio seguente sostituisce l'attuale documento Date con uno nuovo che ha solo i campi `Month` e `Day`, mentre `Year` è stato eliminato.

```
db.example.update(
    { "DocName" : "Document 1" },
```

```
    { $set: { "Date": { "Month": 4, "Day": 18 } } }
)
```

L'aspetto dei risultati di questa operazione è simile al seguente.

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

Inserimento di nuovi campi in un documento incorporato

Per aggiungere campi a un documento incorporato

Per modificare un documento aggiungendo uno o più campi nuovi a un documento incorporato, utilizza l'operazione `update()` con i documenti che identificano il documento incorporato e la "dot notation" per specificare il documento incorporato e i nuovi campi e valori da inserire utilizzando l'operatore `$set`.

Considerato il seguente documento, il codice seguente usa la "dot notation" per inserire i campi `Year` e `DoW` nel documento incorporato `Date` e `Words` nel documento padre.

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

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

L'aspetto dei risultati di questa operazione è simile al seguente.

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

Rimozione di un campo da un documento

Per modificare un documento rimuovendo un campo, utilizza l'operazione `update()` con un documento di query che identifica il documento da cui rimuovere il campo e l'operatore `$unset` per specificare il campo da rimuovere.

L'esempio seguente rimuove il campo `Words` dal documento precedente.

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

L'aspetto dei risultati di questa operazione è simile al seguente.

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

Rimozione di un campo da più documenti

Per modificare un documento rimuovendo un campo da più documenti, utilizza l'operazione `update()` con l'operatore `$unset` e l'opzione `multi` impostata su `true`.

L'esempio seguente rimuove il campo `Inventory` da tutti i documenti della raccolta di esempi. Se un documento non contiene il campo `Inventory`, non viene eseguita alcuna azione. Se `multi: true` viene omesso, l'azione viene eseguita solo nel primo documento che soddisfa il criterio.

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

Eliminazione di documenti

Per rimuovere un documento dal database, utilizza l'operazione `remove()`, specificando il documento da rimuovere. Il codice seguente rimuove "Gel Pen" dalla raccolta `example`.

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

Per rimuovere tutti i documenti dal database, utilizzate l'`remove()`operazione con una query vuota.

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

Inizia a usare Amazon DocumentDB

Esistono molti modi per connettersi e iniziare a usare Amazon DocumentDB. Questa guida è il modo più rapido, semplice e facile per gli utenti di iniziare a utilizzare il nostro potente database di documenti. Questa guida consente AWS CloudShell di connettere e interrogare il cluster Amazon DocumentDB direttamente da. Console di gestione AWS I nuovi clienti idonei per il piano AWS gratuito possono utilizzare Amazon DocumentDB e CloudShell gratuitamente. Se il tuo AWS CloudShell ambiente o il cluster Amazon DocumentDB utilizza risorse oltre il livello gratuito, ti verranno addebitate le AWS tariffe normali per tali risorse. Questa guida ti aiuterà a iniziare a usare Amazon DocumentDB in meno di cinque minuti.

> **Note**
>
> Le istruzioni contenute in questa guida riguardano specificamente la creazione e la connessione a cluster basati su istanze Amazon DocumentDB in cui sono disponibili Amazon DocumentDB. AWS CloudShell
>
> - Se desideri creare e connetterti a cluster elastici Amazon DocumentDB, consulta. Inizia a usare i cluster elastici Amazon DocumentDB
> - Se ti trovi nelle regioni AWS della Cina, consulta. Connetti Amazon EC2 automaticamente

Argomenti

- Prerequisiti
- Fase 1: creazione di un cluster
- Fase 2: Connettiti al tuo cluster
- Fase 3: Inserimento ed interrogazione dei dati
- Fase 4: Esplora
- Guida rapida all'utilizzo di Amazon DocumentDB CloudFormation

Prerequisiti

Prima di creare il tuo primo cluster Amazon DocumentDB, devi fare quanto segue:

Crea un account Amazon Web Services (AWS)

Prima di poter iniziare a utilizzare Amazon DocumentDB, devi disporre di un account Amazon Web Services (AWS). L' AWS account è gratuito. Paghi solo per i servizi e le risorse che utilizzi.

Se non ne hai uno Account AWS, completa i seguenti passaggi per crearne uno.

Per iscriverti a Account AWS

1. Aperta https://signin.aws.amazon.com/signup? request_type=registrati.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata o un messaggio di testo e ti verrà chiesto di inserire un codice di verifica attraverso la tastiera del telefono.

   Quando ti iscrivi a un Account AWS, viene creato un. Utente root dell'account AWS L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

Configura le autorizzazioni AWS Identity and Access Management (IAM) necessarie.

L'accesso per gestire le risorse di Amazon DocumentDB come cluster, istanze e gruppi di parametri del cluster richiede credenziali che AWS possono essere utilizzate per autenticare le tue richieste. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per Amazon DocumentDB.

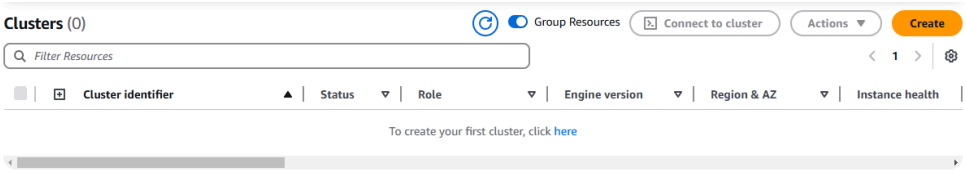
1. Nella barra di ricerca di Console di gestione AWS, inserisci **`IAM`** e seleziona IAM nel menu a discesa che appare.
2. Una volta entrati nella console IAM, seleziona Utenti dal pannello di navigazione.
3. Seleziona il tuo nome utente.
4. Scegli Add Permissions (Aggiungi autorizzazioni).
5. Seleziona Allega direttamente le politiche.
6. `AmazonDocDBFullAccess`Digita la barra di ricerca e selezionala una volta visualizzata nei risultati della ricerca.
7. Scegli Avanti.
8. Scegli Add Permissions (Aggiungi autorizzazioni).

> **Note**
>
> Il tuo AWS account include un VPC predefinito in ogni regione. Se scegli di utilizzare un Amazon VPC, completa i passaggi nell'argomento Creare un Amazon VPC nella Amazon VPC User Guide.

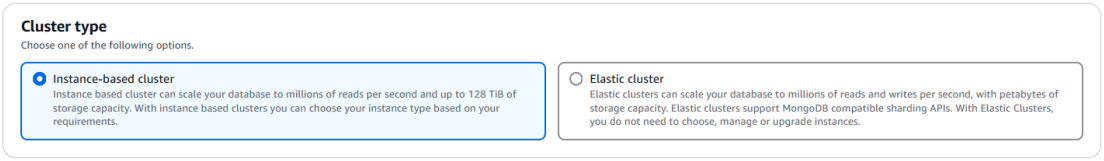
Fase 1: creazione di un cluster

In questo passaggio creerai un cluster Amazon DocumentDB.

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nella console di gestione Amazon DocumentDB, in Clusters, scegli Crea.



3. Nella pagina Crea cluster Amazon DocumentDB, nella sezione Tipo di cluster, scegli Instance-based cluster (questa è l'opzione predefinita).

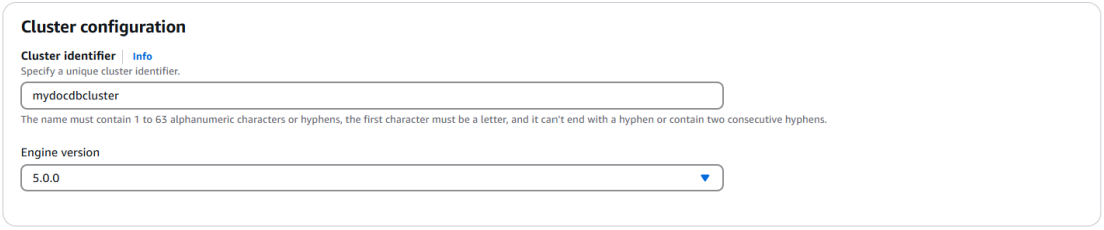


> **Note**
>
> L'altra opzione in questa categoria è Elastic cluster. Per ulteriori informazioni sui cluster elastici Amazon DocumentDB, consulta Utilizzo di Amazon DocumentDB Elastic Clusters

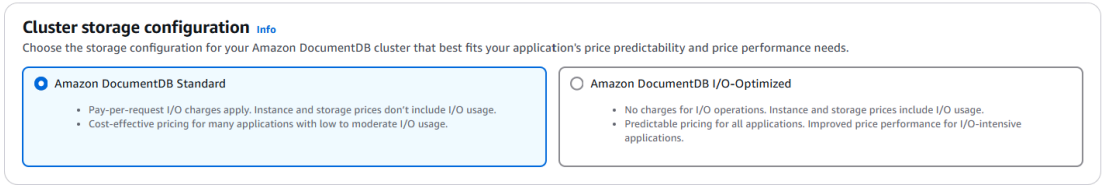
4. Nella sezione di configurazione del cluster:

   a. Per l'identificatore del cluster, inserisci un nome univoco, ad esempio**`mydocdbcluster`**. Tieni presente che la console cambierà tutti i nomi dei cluster in lettere minuscole indipendentemente da come vengono immessi.

   b. Per la versione Engine, scegli 5.0.0.



5. Nella sezione di configurazione dello storage del cluster, scegli Amazon DocumentDB Standard (questa è l'opzione predefinita).



> **Note**
>
> L'altra opzione in questa categoria è Amazon I/O-Optimized DocumentDB. Per ulteriori informazioni su entrambe le opzioni, consulta Configurazioni di storage del cluster Amazon DocumentDB

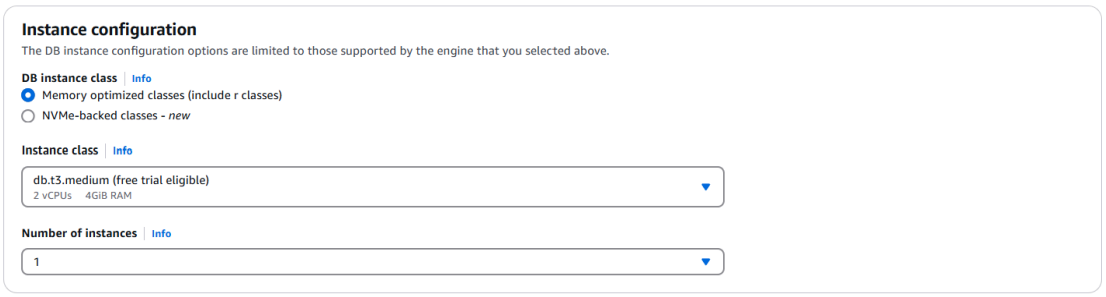
6. Nella sezione Configurazione dell'istanza:

   a. Per la classe di istanza DB, scegli le classi ottimizzate per la memoria (includi le classi r) (impostazione predefinita).

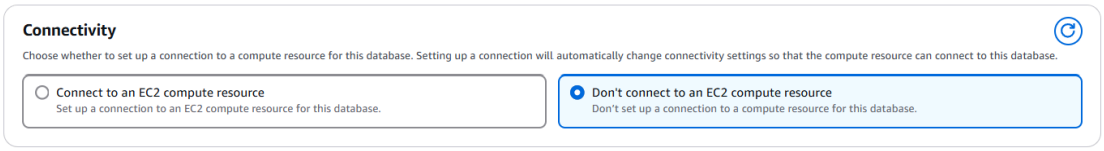
      L'altra opzione di istanza sono NVMe-backed le classi. Per ulteriori informazioni, consulta NVMe-backed istanze.

   b. Per la classe Instance, scegli db.t3.medium. Questa è idonea per la prova gratuita. AWS

   c. Per Numero di istanze, scegli 1 istanza. La scelta di un'istanza aiuta a minimizzare i costi. Se si trattasse di un sistema di produzione, consigliamo di effettuare il provisioning di tre istanze per garantire un'elevata disponibilità.

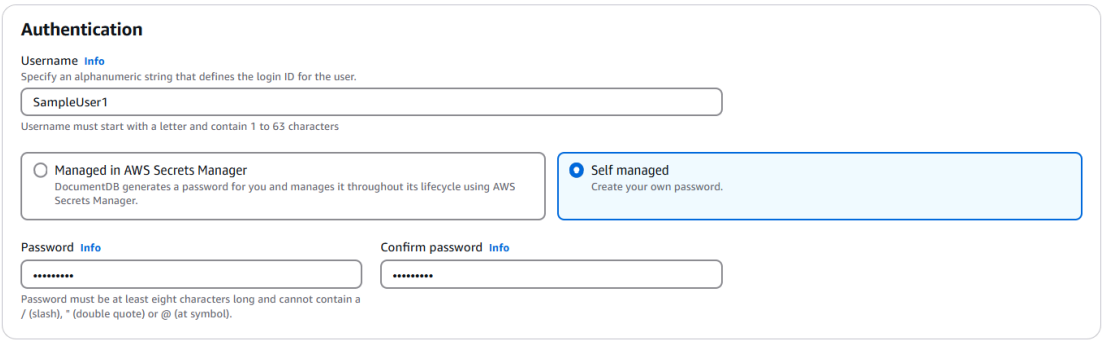


7. Nella sezione Connettività, lascia l'impostazione predefinita di Non connetterti a una risorsa di calcolo EC2.



8. Nella sezione Autenticazione, inserisci un nome utente per l'utente principale, quindi scegli Autogestito. Inserisci una password, quindi confermala.

   Se invece hai scelto Managed in AWS Secrets Manager, consulta Gestione delle password con Amazon DocumentDB e AWS Secrets Manager per ulteriori informazioni.



9. Lascia tutte le altre opzioni come predefinite e scegli Crea cluster.

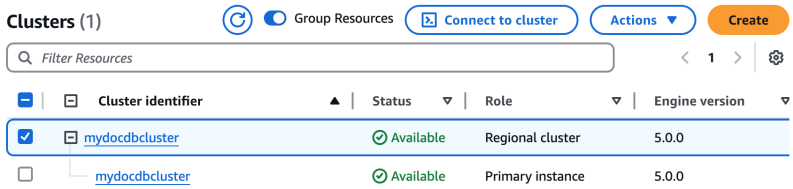
Amazon DocumentDB sta ora effettuando il provisioning del cluster, operazione che può richiedere alcuni minuti per essere completata.

> **Note**
>
> Per informazioni sui valori di stato del cluster, consulta Valori di stato del cluster il capitolo Monitoraggio di Amazon DocumentDB.

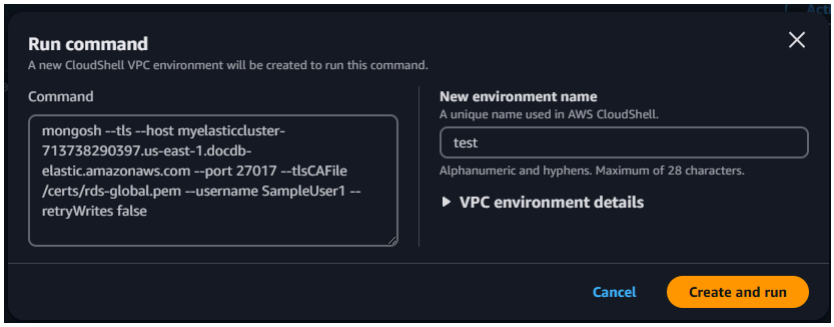
Fase 2: Connettiti al tuo cluster

Connettiti al tuo cluster Amazon DocumentDB utilizzando AWS CloudShell.

1. Nella console di gestione Amazon DocumentDB, in Clusters, individua il cluster che hai creato. Scegli il tuo cluster facendo clic sulla casella di controllo accanto ad esso.



2. Scegli Connetti al cluster (che si trova accanto al menu a discesa Azioni). Questo pulsante è abilitato solo dopo aver fatto clic sulla casella di controllo accanto al cluster e lo stato del cluster regionale e delle istanze primarie viene visualizzato come Disponibile. Viene visualizzata la schermata del comando CloudShell Esegui.
3. Nel campo Nome nuovo ambiente, inserisci un nome univoco, ad esempio «test» e scegli Crea ed esegui. I dettagli dell'ambiente VPC vengono configurati automaticamente per il database Amazon DocumentDB.



4. Quando richiesto, inserisci la password che hai creato nella fase 1: creazione di un cluster Amazon DocumentDB (sottofase 7).



   Una volta inserita la password e ricevuta la richiesta`rs0 [direct: primary] <env-name>>`, la connessione al cluster Amazon DocumentDB è stata completata.

> **Note**
>
> Per informazioni sulla risoluzione dei problemi, consulta Troubleshooting Amazon DocumentDB.

Fase 3: Inserimento ed interrogazione dei dati

Ora che sei connesso al tuo cluster, puoi eseguire alcune query per acquisire familiarità con l'utilizzo di un database di documenti.

1. Per inserire un singolo documento, inserisci quanto segue:

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

   Si ottiene il seguente risultato:

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

2. Puoi leggere il documento che hai scritto con il `findOne()` comando (perché restituisce solo un singolo documento). Inserisci quanto segue:

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

   Si ottiene il seguente risultato:

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

3. Per eseguire qualche altra domanda, considera un caso d'uso dei profili di gioco. Innanzitutto, inserisci alcune voci in una raccolta intitolata`profiles`. Inserisci quanto segue:

   ```
   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 }
   ```

```
])
```

Si ottiene il seguente risultato:

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

4. Usa il `find()` comando per restituire tutti i documenti nella raccolta dei profili. Inserisci quanto segue:

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

Otterrai un risultato che corrisponderà ai dati che hai digitato nel passaggio 3.

5. Usa una query per un singolo documento usando un filtro. Inserisci quanto segue:

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

Si ottiene il seguente risultato:

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

6. Ora proviamo a trovare un profilo e modificarlo usando il `findAndModify` comando. Daremo all'utente Matt altri 10 punti con il seguente codice:

```
db.profiles.findAndModify({
        query: { name: "Matt", status: "active"},
        update: { $inc: { score: 10 } }
    })
```

Si ottiene il seguente risultato (nota che il suo punteggio non è ancora aumentato):

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

7. Puoi verificare che il suo punteggio sia cambiato con la seguente domanda:

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

Si ottiene il seguente risultato:

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

Fase 4: Esplora

Complimenti! Hai completato con successo la guida introduttiva per i cluster basati su istanze Amazon DocumentDB.

Qual è il prossimo passo? Scopri come sfruttare appieno questo database con alcune delle sue funzionalità più popolari:

- Gestione di Amazon DocumentDB
- Dimensionamento
- Backup e ripristino

> **Note**
>
> Il cluster creato con questo esercizio introduttivo continuerà ad accumulare costi a meno che non venga eliminato. Per istruzioni, consulta Eliminazione di un cluster Amazon DocumentDB.

Guida rapida all'utilizzo di Amazon DocumentDB CloudFormation

Questa sezione contiene passaggi e altre informazioni per aiutarti a iniziare rapidamente con Amazon DocumentDB (con compatibilità MongoDB) utilizzando. CloudFormation Per informazioni generali su Amazon DocumentDB, consulta. Cos'è Amazon DocumentDB (con compatibilità con MongoDB)

Queste istruzioni utilizzano un AWS CloudFormation modello per creare un cluster e istanze nel tuo Amazon VPC predefinito. Per istruzioni su come creare queste risorse, consultare Inizia a usare Amazon DocumentDB.

> **Important**
>
> Lo CloudFormation stack creato da questo modello crea più risorse, incluse le risorse in Amazon DocumentDB (ad esempio, un cluster e istanze) e Amazon Elastic Compute Cloud (ad esempio, un gruppo di sottorete).

Alcune di queste risorse non sono gratuite. Per informazioni sui prezzi, consulta i prezzi di Amazon DocumentDB e i prezzi di Amazon EC2. https://aws.amazon.com/ec2/pricing/ Al termine, è possibile eliminare lo stack per interrompere gli addebiti.

Questo CloudFormation stack è destinato esclusivamente a scopi didattici. Se utilizzi questo modello per un ambiente di produzione, utilizza politiche e sicurezza IAM più rigorose. Per informazioni sulla protezione delle risorse, consulta Amazon VPC Security e Amazon EC2 Network and Security.

Argomenti

- Prerequisiti
- Avvio di un Amazon DocumentDB CloudFormation stack
- Accesso al cluster Amazon DocumentDB
- Protezione dalla terminazione e protezione dall'eliminazione

Prerequisiti

Prima di creare un cluster Amazon DocumentDB, devi disporre di quanto segue:

- Un Amazon VPC predefinito
- Le autorizzazioni IAM richieste

Autorizzazioni IAM richieste

Le seguenti autorizzazioni consentono di creare le risorse per lo stack CloudFormation :

AWS Politiche gestite

- `AWSCloudFormationReadOnlyAccess`
- `AmazonDocDBFullAccess`

Autorizzazioni aggiuntive per IAM

La seguente politica delinea le autorizzazioni aggiuntive necessarie per creare ed eliminare questo CloudFormation stack.

Negli esempi seguenti, sostituisci ciascuno *user input placeholder* con le informazioni della tua risorsa.

Coppia di chiavi Amazon EC2

È necessario disporre di una coppia di chiavi (e del file PEM) disponibili nella regione in cui verrà creato lo CloudFormation stack. Se devi creare una coppia di chiavi, consulta Creating a Key Pair Using Amazon EC2 nella Amazon EC2 User Guide.

Avvio di un Amazon DocumentDB CloudFormation stack

Questa sezione descrive come avviare e configurare uno stack Amazon CloudFormation DocumentDB.

1. Accedi all' Console di gestione AWS indirizzo. https://console.aws.amazon.com/
2. La tabella seguente elenca i modelli di stack Amazon DocumentDB per ciascuno Regione AWS. Scegli Launch Stack per il Regione AWS tipo in cui desideri avviare il tuo stack.

Region	Visualizza modello	Visualizzazione in Designer	Avvia
Stati Uniti orientali (Ohio)	Visualizza modello	Visualizzazione in Designer	
Stati Uniti orientali (Virginia settentrionale)	Visualizza modello	Visualizzazione in Designer	
Stati Uniti occidentali (Oregon)	Visualizza modello	Visualizzazione in Designer	
Asia Pacifico (Mumbai)	Visualizza modello	Visualizzazione in Designer	
Asia Pacifico (Seul)	Visualizza modello	Visualizzazione in Designer	
Asia Pacifico (Singapore)	Visualizza modello	Visualizzazione in Designer	

Region	Visualizza modello	Visualizzazione in Designer	Avvia
Asia Pacifico (Sydney)	Visualizza modello	Visualizzazione in Designer	Launch Stack
Asia Pacifico (Tokyo)	Visualizza modello	Visualizzazione in Designer	Launch Stack
Canada (Centrale)	Visualizza modello	Visualizzazione in Designer	Launch Stack
Europa (Francoforte)	Visualizza modello	Visualizzazione in Designer	Launch Stack
Europa (Irlanda)	Visualizza modello	Visualizzazione in Designer	Launch Stack
Europa (Londra)	Visualizza modello	Visualizzazione in Designer	Launch Stack
Europa (Parigi)	Visualizza modello	Visualizzazione in Designer	Launch Stack

3. Crea stack: descrive il modello Amazon DocumentDB selezionato. Ogni stack è basato su un modello, un file JSON o YAML, che contiene la configurazione delle AWS risorse da includere nello stack. Poiché hai scelto di avviare uno stack dai modelli forniti sopra, il modello è già stato configurato per creare uno stack Amazon DocumentDB per quello che hai scelto. Regione AWS

   Quando avvii uno CloudFormation stack, la protezione dall'eliminazione per il tuo cluster Amazon DocumentDB è disabilitata per impostazione predefinita. Se si desidera abilitare la protezione da eliminazione per il cluster, eseguire la procedura seguente. In caso contrario, scegliere Next (Avanti) per continuare con la fase successiva.

   Per abilitare la protezione dall'eliminazione per il tuo cluster Amazon DocumentDB:

   1. Scegliere View in Designer (Visualizza in Designer) nell'angolo in basso a destra della pagina Create stack (Crea stack) .
   2. Modifica il modello utilizzando l'editor JSON e YAML integrato nella pagina CloudFormation Designer risultante della console. Scorrere fino alla sezione `Resources` e modificala

per includere `DeletionProtection`, come segue. Per ulteriori informazioni sull'uso di CloudFormation Designer, consulta What Is Designer? CloudFormation.

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. Scegliere Create Stack (Crea stack) () nell'angolo in alto a sinistra della pagina per salvare le modifiche e creare uno stack con queste modifiche abilitate.
4. Dopo aver salvato le modifiche, si verrà reindirizzati alla pagina Create Stack (Crea stack) .
5. Seleziona Successivo per continuare.

4. Specifica i dettagli dello stack: inserisci il nome dello stack e i parametri per il tuo modello. I parametri sono definiti nel modello e consentono di immettere valori personalizzati quando crei o aggiorni uno stack.

   - In Stack name (Nome stack), immettere un nome per lo stack o accettare il nome fornito. Il nome dello stack può includere lettere (A—Z e a—z), numeri (0—9) e trattini (—).
   - In Parameters (Parametri), immettere i seguenti dettagli:

     - DBClusterName— Inserisci un nome per il tuo cluster Amazon DocumentDB o accetta il nome fornito.

       Vincoli per la denominazione del cluster:

       - La lunghezza è di [1—63] lettere, numeri o trattini.
       - Il primo carattere deve essere una lettera.
       - Non può terminare con un trattino o contenere due trattini consecutivi.
       - Deve essere univoco per tutti i cluster di Amazon RDS, Neptune e Amazon DocumentDB per regione. Account AWS

     - DBInstanceClass— Dall'elenco a discesa, seleziona la classe di istanza per il tuo cluster Amazon DocumentDB.
     - DBInstanceName— Inserisci un nome per la tua istanza Amazon DocumentDB o accetta il nome fornito.

       Vincoli per la denominazione di un'istanza:

       - La lunghezza è di [1—63] lettere, numeri o trattini.
       - Il primo carattere deve essere una lettera.
       - Non può terminare con un trattino o contenere due trattini consecutivi.
       - Deve essere univoco per tutte le istanze su Amazon RDS, Neptune e Amazon DocumentDB per regione. Account AWS

     - MasterPassword— La password dell'account amministratore del database.
     - MasterUser— Il nome utente dell'account amministratore del database. MasterUser Deve iniziare con una lettera e può contenere solo caratteri alfanumerici.

   Scegliere Next (Avanti) per salvare le modifiche e continuare.

5. Configura le opzioni dello stack: configura i tag, le autorizzazioni e le opzioni aggiuntive dello stack.

- Tag: specifica le coppie di tag (chiave-valore) da applicare alle risorse nello stack. È possibile aggiungere fino a 50 tag univoci per ogni stack.
- Autorizzazioni: facoltative. Scegli un ruolo IAM per definire in modo esplicito come CloudFormation creare, modificare o eliminare le risorse nello stack. Se non scegli un ruolo, CloudFormation utilizza le autorizzazioni basate sulle credenziali dell'utente. Prima di specificare un ruolo del servizio, assicurarsi di disporre dell'autorizzazione per passarlo (`iam:PassRole`). L'autorizzazione `iam:PassRole` specifica quali ruolo puoi utilizzare.

  > **Note**
  >
  > Quando specifichi un ruolo di servizio, utilizza CloudFormation sempre quel ruolo per tutte le operazioni eseguite su quello stack. Gli altri utenti che hanno le autorizzazioni per eseguire operazioni su questo stack saranno in grado di utilizzare questo ruolo, anche se non dispongono dell'autorizzazione per passarlo. Se il ruolo include le autorizzazioni di cui l'utente non dovrebbe disporre, puoi riassegnare involontariamente le autorizzazioni di un utente. Assicurati che il ruolo conceda il privilegio https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege minimo.

- Opzioni avanzate: puoi impostare le seguenti opzioni avanzate:
  - Politica dello stack: opzionale. Definisce le risorse che si desidera proteggere da aggiornamenti involontari durante un aggiornamento dello stack. Per impostazione predefinita, durante un aggiornamento dello stack possono essere aggiornate tutte le risorse.

    È possibile inserire la policy di stack direttamente come JSON o caricare un file in formato JSON che contenga la policy dello stack. Per ulteriori informazioni, consulta Impedire gli aggiornamenti delle risorse stack.
  - Configurazione di rollback: opzionale. Specifica gli allarmi CloudWatch dei log CloudFormation da monitorare durante la creazione e l'aggiornamento dello stack. Se l'operazione supera una soglia di allarme, la CloudFormation ripristina.
  - Opzioni di notifica: facoltative. Specificare argomenti per Simple Notification System (SNS).
  - Opzioni di creazione dello stack: facoltative. Puoi specificare le seguenti opzioni:

- Rollback in caso di errore: indica se lo stack deve essere ripristinato o meno se la creazione dello stack fallisce.
- Timeout: il numero di minuti prima del timeout per la creazione di uno stack.
- Protezione dalla terminazione: impedisce l'eliminazione accidentale dello stack.

> **Note**
>
> CloudFormation la protezione dalla terminazione è diversa dal concetto di protezione da eliminazione di Amazon DocumentDB. Per ulteriori informazioni, consulta Protezione dalla terminazione e protezione dall'eliminazione.

Seleziona Successivo per continuare.

6. Revisione <stack-name>: esamina il modello di stack, i dettagli e le opzioni di configurazione. È inoltre possibile aprire un quick-create link (collegamento di creazione rapida) nella parte inferiore della pagina per creare stack con le stesse configurazioni di base di questo.

   - Scegliere Create (Crea) per creare lo stack.
   - In alternativa, è possibile scegliere Create change set (Crea set di modifiche). Un set di modifiche è un'anteprima di come verrà configurato questo stack prima che venga creato. Ciò consente di esaminare varie configurazioni prima di eseguire il set di modifiche.

Accesso al cluster Amazon DocumentDB

Una volta completato lo CloudFormation stack, puoi utilizzare un'istanza Amazon EC2 per connetterti al tuo cluster Amazon DocumentDB. Per informazioni sulla connessione a un'istanza Amazon EC2 tramite SSH, consulta Connect to Your Linux Instance nella Amazon EC2 User Guide.

Dopo esserti connesso, consulta le seguenti sezioni, che contengono informazioni sull'utilizzo di Amazon DocumentDB.

- Fase 3: Inserimento ed interrogazione dei dati
- Gestione di cluster e istanze Amazon DocumentDB
- Monitoraggio e registrazione in Amazon DocumentDB

Protezione dalla terminazione e protezione dall'eliminazione

È una best practice di Amazon DocumentDB per abilitare la protezione da cancellazione e la protezione dalla cessazione. CloudFormation la protezione dalla terminazione è una funzionalità nettamente diversa dalla funzionalità di protezione dalle eliminazioni di Amazon DocumentDB.

- Protezione dalla terminazione: puoi evitare che uno stack venga eliminato accidentalmente attivando la protezione dalla terminazione per il tuo stack. CloudFormation Se un utente tenta di eliminare uno stack con protezione da cessazione abilitata, l'eliminazione ha esito negativo e lo stack rimane invariato. La protezione dalle terminazioni è disabilitata per impostazione predefinita quando si crea uno stack utilizzando. CloudFormation Puoi abilitare la protezione da cessazione sullo stack quando lo crei. Per ulteriori informazioni, consulta Impostazione delle opzioni CloudFormation dello stack.
- Protezione dall'eliminazione: Amazon DocumentDB offre anche la possibilità di abilitare la protezione dall'eliminazione per un cluster. Se un utente tenta di eliminare un cluster Amazon DocumentDB con la protezione dall'eliminazione abilitata, l'eliminazione non riesce e il cluster rimane invariato. La protezione dalle eliminazioni, se abilitata, protegge dalle eliminazioni accidentali da Amazon DocumentDB e. Console di gestione AWS AWS CLI CloudFormation Per ulteriori informazioni sull'attivazione e la disabilitazione della protezione da eliminazione per un cluster Amazon DocumentDB, consulta. Protezione da eliminazione

Migrazione e aggiornamento di Amazon DocumentDB

Amazon DocumentDB è un servizio di database completamente gestito compatibile con l'API MongoDB. Puoi migrare i tuoi dati su Amazon DocumentDB dai database MongoDB in esecuzione in locale o su Amazon Elastic Compute Cloud (Amazon EC2) utilizzando la procedura descritta in questa sezione.

Argomenti

- Esegui la migrazione ad Amazon DocumentDB utilizzando AWS Database Migration Service (DMS): guida rapida
- Runbook sulla migrazione di Amazon DocumentDB
- Migrazione da Couchbase Server
- Date di supporto delle versioni del motore Amazon DocumentDB
- Aggiornamento della versione principale in-place di Amazon DocumentDB
- Aggiornamento della versione secondaria di Amazon DocumentDB
- Aggiornamento del cluster Amazon DocumentDB tramite AWS Database Migration Service
- Uso di versioni con supporto a lungo termine (Long-Term Support, LTS)
- Supporto esteso Amazon DocumentDB

Esegui la migrazione ad Amazon DocumentDB utilizzando AWS Database Migration Service (DMS): guida rapida

Argomenti

- Prepara la fonte DMS
- Configura DMS
- Abilita la compressione DocumentDB
- Crea un'attività di replica
- Monitoraggio dell'avanzamento
- Informazioni aggiuntive

Prepara la fonte DMS

Vedi Abilitazione dei flussi di modifica per abilitare i flussi di modifica di DocumentDB o abilitare l'Oplog MongoDB per supportare DMS Change Data Capture (CDC).

- La fonte DMS deve conservare tutte le modifiche in corso fino al completamento del caricamento completo del DMS per tutte le raccolte incluse.
- I flussi di modifiche di DocumentDB sono basati sul tempo. Assicurati che le tue `change_stream_log_retention_duration` impostazioni siano sufficientemente grandi da coprire il tempo necessario per completare il caricamento completo.
- L'oplog MongoDB ha una dimensione fissa. Assicurati che sia dimensionato per contenere tutte le operazioni a pieno carico.

Configura DMS

Crea un'istanza DMS, gli endpoint di origine e di destinazione e testa ogni endpoint.

Abilita la compressione DocumentDB

Amazon DocumentDB 5.0: abilita la compressione collegando un gruppo di parametri personalizzato al cluster Amazon DocumentDB e impostando il parametro su. `default_collection_compression` `enabled` Per ulteriori informazioni, consulta Gestione della compressione dei documenti a livello di raccolta.

Amazon DocumentDB 8.0: la compressione Zstd è abilitata per impostazione predefinita. Per ulteriori informazioni, consulta Gestione della compressione basata su dizionari in Amazon DocumentDB 8.0.

Crea un'attività di replica

1. Nella console DMS, nel pannello di navigazione, scegli Migra o replica, quindi scegli Attività.
2. Scegli Create task (Crea attività).
3. Nella pagina Crea attività, nella sezione Configurazione dell'attività:
   - Inserisci un identificatore di attività univoco e significativo (ad esempio, «mongodb-docdb-replication»).
   - Scegli l'endpoint di origine che hai creato in precedenza nel menu a discesa Endpoint del database di origine.

- Scegli l'endpoint di destinazione che hai creato in precedenza nel menu a discesa Endpoint del database di Target.
- Per Tipo di attività, scegli Migra e replica.

4. Nella sezione Impostazioni:

- Per i registri delle attività, seleziona la casella Attiva CloudWatch i registri.
- Per la modalità di modifica (nella parte superiore della sezione), scegli l'editor JSON e imposta i seguenti attributi:
  - Impostato `ParallelApplyThreads` su 5 (sotto`TargetMetadata`). Ciò consente ~1000 operazioni insert/update di cancellazione al secondo in CDC.
  - Impostato su `MaxFullLoadSubTasks` 16 (inferiore). `FullLoadSettings` Valuta la possibilità di aumentarlo in base alle dimensioni dell'istanza.
  - Per raccolte di grandi dimensioni (oltre 100 GB), abilita la partizione automatica (in Table Mapping e sotto l'`parallel-load`attributo):
    - «type»: «partitions-auto»
    - «numero di partizioni»: 16

Monitoraggio dell'avanzamento

Usa la AWS DMS console o crea una dashboard personalizzata (strumento dashboard) per tenere traccia della migrazione. Concentrati sulle seguenti metriche:

- FullLoadThroughputBandwidthTarget— Misura la larghezza di banda di rete (in KB/second) utilizzata dal DMS durante il trasferimento dei dati al database di destinazione durante la fase di migrazione a pieno carico.
- CDCLatencyTarget— Misura il ritardo (in secondi) tra il momento in cui si verifica una modifica nel database di origine e quando tale modifica viene applicata al database di destinazione.
- CDCThroughputRowsTarget— Misura il numero di righe al secondo che DMS applica al database di destinazione durante la fase di replica in corso della migrazione.

Informazioni aggiuntive

Per ulteriori informazioni su Amazon DocumentDB e AWS DMS, consulta:

- Runbook sulla migrazione di Amazon DocumentDB

- Migrazione da MongoDB ad Amazon DocumentDB

Runbook sulla migrazione di Amazon DocumentDB

Questo runbook fornisce una guida completa per la migrazione di un database MongoDB su Amazon DocumentDB utilizzando (DMS). AWS Database Migration Service È progettato per supportare amministratori di database, ingegneri del cloud e sviluppatori durante tutto il percorso di migrazione end-to-end, dalla scoperta iniziale alla convalida successiva alla convalida successiva alla migrazione.

Date le differenze di implementazione e funzionalità supportate tra MongoDB e Amazon DocumentDB, questo runbook enfatizza un approccio strutturato e sistematico. Descrive le valutazioni essenziali prima della migrazione, evidenzia le considerazioni sulla compatibilità e descrive in dettaglio le attività chiave necessarie per garantire una migrazione di successo con interruzioni minime.

Il runbook è organizzato nei seguenti argomenti:

- Compatibilità— Comprendi le funzionalità e i tipi di dati di MongoDB supportati in Amazon DocumentDB e identifica le potenziali incompatibilità.
- Rilevamento dei carichi di lavoro— Analizza i carichi di lavoro MongoDB esistenti, inclusi read/write modelli, volumi di dati e linee di base delle prestazioni.
- Migrazione dell'indice— Analizza le strategie per estrarre e trasformare gli indici MongoDB per ottenere prestazioni ottimali in Amazon DocumentDB.
- Migrazione degli utenti— Descrivi in dettaglio l'approccio per la migrazione di utenti, ruoli e controlli di accesso del database ad Amazon DocumentDB.
- Migrazione dei dati— Scopri vari metodi per la migrazione dei dati utilizzando AWS DMS, tra cui l'acquisizione dei dati a pieno carico e modifica (CDC).
- Monitoraggio— Descrivi in dettaglio i vari approcci di monitoraggio durante la migrazione tramite DMS o strumenti nativi.
- Convalida— Fornire procedure per i controlli dell'integrità dei dati, la convalida funzionale e il confronto delle prestazioni dopo la migrazione.

Seguendo le indicazioni contenute in questo runbook, i team possono garantire una transizione fluida, sicura ed efficiente ad Amazon DocumentDB, preservando al contempo la funzionalità delle applicazioni e riducendo al minimo i rischi.

Compatibilità

Argomenti

- Compatibilità delle funzionalità principali
- Strumento di valutazione della compatibilità di Amazon DocumentDB

Quando si esegue la migrazione da MongoDB ad Amazon DocumentDB, una valutazione iniziale approfondita e un controllo della compatibilità delle funzionalità sono essenziali per una migrazione di successo. Questo processo inizia con un inventario completo delle funzionalità di MongoDB, inclusi operatori di pipeline di aggregazione, modelli di query, indici e modelli di dati.

Poiché Amazon DocumentDB è compatibile con le API MongoDB 3.6, 4.0, 5.0 e 8.0, le applicazioni che utilizzano funzionalità più recenti potrebbero richiedere il refactoring. MongoDB-specific Le aree critiche da valutare includono i meccanismi di sharding (Amazon DocumentDB utilizza un approccio diverso), le implementazioni delle transazioni, le funzionalità dei flussi di modifiche e i tipi di indici (in particolare gli indici sparsi e parziali).

Anche le caratteristiche prestazionali sono diverse, con Amazon DocumentDB ottimizzato per carichi di lavoro aziendali con prestazioni prevedibili. I test dovrebbero comportare l'esecuzione di carichi di lavoro rappresentativi su entrambi i sistemi per identificare i modelli di query che potrebbero richiedere l'ottimizzazione.

Il monitoraggio dei piani di esecuzione per rilevare potenziali lacune prestazionali è importante durante la fase di valutazione. Questo aiuta a creare una chiara tabella di marcia per la migrazione, a identificare le modifiche necessarie alle applicazioni e a stabilire tempistiche realistiche per una transizione senza intoppi.

Compatibilità delle funzionalità principali

Supporto completo delle funzionalità

- Operazioni CRUD: usufruisci del supporto completo per tutte le operazioni di base di creazione, lettura, aggiornamento ed eliminazione, inclusi gli operatori di blocco e di interrogazione, garantendo una perfetta compatibilità delle applicazioni.
- Funzionalità di indicizzazione avanzate: sfrutta il supporto completo per gli indici a campo singolo, composto, TTL, parziale, sparse e 2dsphere, per ottimizzare le prestazioni delle query e gli indici di testo (versione 5) per le ricerche basate su testo.

- Enterprise-grade replica: approfitta di un robusto meccanismo di failover automatico con repliche di lettura per una disponibilità elevata e senza costi operativi.
- Soluzioni di backup avanzate: stai tranquillo con un sistema di backup automatico dotato di Point-in-Time ripristino (PITR) e istantanee manuali su richiesta per la protezione dei dati.

Migliorate AWS-funzionalità integrate

- Aggregazione semplificata: sfrutta le fasi di aggregazione più comunemente utilizzate (`$match`,,, `$group` `$sort``$project`, ecc.) con prestazioni ottimizzate per i carichi di lavoro aziendali.
- Supporto alle transazioni: implementa transazioni con più documenti e più raccolte, perfette per la maggior parte delle esigenze delle applicazioni aziendali.
- Real-time tracciamento dei dati: abilita i flussi di modifiche con un semplice comando e aumenta il periodo di conservazione del flusso di modifiche tramite una semplice impostazione di gruppo di parametri per il monitoraggio delle modifiche ai dati in tempo reale.
- Location-based servizi — Implementazione di applicazioni geospaziali con supporto per `$geoNear` operatori e indici 2dsphere.
- Funzionalità di ricerca testuale: utilizza la funzionalità di ricerca testuale integrata per le esigenze di scoperta dei contenuti.

Vantaggi dell'architettura moderna

- Cloud-native design: usufruisci di un'architettura AWS ottimizzata che sostituisce le funzionalità precedenti, ad esempio MapReduce con operazioni di pipeline di aggregazione più efficienti.
- Sicurezza avanzata: approfitta dell'autenticazione AWS Identity and Access Management (IAM), SCRAM-SHA-1 SCRAM-SHA-256, con X.509 certificato e dell'autenticazione basata su password.
- Prestazioni prevedibili: ottieni prestazioni costanti ottimizzate specificamente per i carichi di lavoro aziendali.

Per una panoramica completa delle funzionalità di Amazon DocumentDB, consulta API, operazioni e tipi di dati MongoDB supportati in Amazon DocumentDB e per Differenze funzionali: Amazon DocumentDB e MongoDB massimizzare il potenziale del database.

Amazon DocumentDB non supporta tutti gli indici offerti da MongoDB. Forniamo uno strumento di indicizzazione gratuito per verificare la compatibilità. GitHub Esegui lo strumento di indicizzazione per valutare l'incompatibilità e pianificare le soluzioni alternative di conseguenza.

Strumento di valutazione della compatibilità di Amazon DocumentDB

Lo strumento di compatibilità di Amazon DocumentDB GitHub analizza il MongoDB carico di lavoro e riporta operatori, comandi e funzionalità che non sono supportati o richiedono modifiche in Amazon DocumentDB. Supporta le versioni target dalla 3.6 alla 8.0.1 (impostazione predefinita: 8.0.1).

Consigliata: modalità URI (5.0+) MongoDB

La modalità URI si connette direttamente all'istanza MongoDB, campiona le operazioni in tempo reale e non richiede profilazione o accesso ai log.

Metodi alternativi

- Log-based analisi: analizza i log del MongoDB profiler. Acquisisce il comportamento effettivo in fase di esecuzione ma richiede l'abilitazione della profilazione e copre solo il periodo di registrazione.
- Analisi del codice sorgente: analizza il codice sorgente dell'applicazione per l'utilizzo dell'API. MongoDB Fornisce una copertura completa ma può contrassegnare i percorsi di codice inutilizzati e non è in grado di rilevare le query costruite dinamicamente.

Prerequisiti

- Python3.7+
- Clona lo strumento: `git clone https://github.com/awslabs/amazon-documentdb-tools.git`
- Installa le dipendenze: `pip install -r compat-tool/requirements.txt`

Per i dettagli completi sull'utilizzo e tutte le opzioni disponibili, consulta il file README su. GitHub

Rilevamento dei carichi di lavoro

La migrazione da MongoDB ad Amazon DocumentDB richiede una conoscenza approfondita del carico di lavoro del database esistente. Il rilevamento del carico di lavoro è il processo di analisi dei modelli di utilizzo del database, delle strutture dei dati, delle prestazioni delle query e delle dipendenze operative per garantire una transizione senza interruzioni con interruzioni minime. Questa sezione illustra i passaggi chiave coinvolti nell'individuazione dei carichi di lavoro per facilitare una migrazione efficace da MongoDB ad Amazon DocumentDB.

Argomenti

- Valutazione della distribuzione MongoDB esistente
- Identificazione delle differenze tra i modelli di dati
- Analisi delle query e delle prestazioni
- Revisione della sicurezza e del controllo degli accessi
- Considerazioni operative e di monitoraggio

Valutazione della distribuzione MongoDB esistente

Prima della migrazione, è fondamentale valutare l'attuale ambiente MongoDB, tra cui:

- Architettura del cluster: identifica il numero di nodi, i set di repliche e le configurazioni di sharding. Quando si esegue la migrazione da MongoDB ad Amazon DocumentDB, è importante comprendere la configurazione dello sharding MongoDB perché Amazon DocumentDB non supporta lo sharding controllato dall'utente. Le applicazioni progettate per un ambiente MongoDB frammentato richiederanno modifiche all'architettura, poiché Amazon DocumentDB utilizza un approccio di scalabilità diverso con la sua architettura basata sullo storage. Dovrai adattare la tua strategia di distribuzione dei dati e possibilmente consolidare le raccolte frammentate quando passi ad Amazon DocumentDB.
- Storage e volume di dati: misura la dimensione totale dei dati e la dimensione dell'indice del cluster. Completalo con lo strumento di revisione Oplog per comprendere i modelli di scrittura e la velocità di crescita dei dati. Per ulteriori informazioni sul dimensionamento del cluster, consulta. Dimensionamento delle istanze
- Schemi di carico di lavoro: analizza il throughput di lettura e scrittura, la frequenza di esecuzione delle query e l'efficienza di indicizzazione.
- Dipendenze operative: documenta tutte le applicazioni, i servizi e le integrazioni che si basano su MongoDB.

Identificazione delle differenze tra i modelli di dati

Sebbene Amazon DocumentDB lo sia MongoDB-compatible, ci sono differenze nelle funzionalità supportate, come:

- Transazioni: Amazon DocumentDB supporta le transazioni ACID, ma con alcune. Limitazioni

- Progettazione dello schema: assicurati che le strutture dei documenti, i documenti incorporati e i riferimenti siano in linea con le best practice di Amazon DocumentDB.

Analisi delle query e delle prestazioni

La comprensione del comportamento delle query aiuta a ottimizzare le prestazioni di migrazione e post-migrazione. Le aree chiave da analizzare includono:

- Query lente: identifica le query con tempi di esecuzione elevati utilizzando gli strumenti di profilazione di MongoDB.
- Schemi di interrogazione: categorizza i tipi di query più comuni, comprese le operazioni e le aggregazioni CRUD.
- Utilizzo dell'indice: valuta se gli indici sono utilizzati in modo efficace o necessitano di ottimizzazione in Amazon DocumentDB. Per valutare l'utilizzo degli indici e ottimizzare le prestazioni in Amazon DocumentDB, utilizza la fase della pipeline di `$indexStats` aggregazione combinata con il metodo nelle tue query critiche. `explain()` Inizia eseguendo l'esecuzione `db.collection.aggregate([{$indexStats{}}])` per identificare quali indici vengono utilizzati. Puoi eseguire un'analisi più dettagliata eseguendo le query più frequenti con. `explainPlan`
- Concorrenza e distribuzione del carico di lavoro: valuta i rapporti di lettura e scrittura, il pool di connessioni e i colli di bottiglia delle prestazioni.

Revisione della sicurezza e del controllo degli accessi

Autenticazione e autorizzazione

- MongoDB RBAC su Amazon DocumentDB IAM e RBAC: mappa gli utenti e i ruoli di controllo degli accessi basati sui ruoli di MongoDB alle policy (IAM) e agli utenti di autenticazione SCRAM di Amazon DocumentDB. AWS Identity and Access Management
- Strategia di migrazione degli utenti: piano per la migrazione degli utenti del database, dei ruoli personalizzati e dei privilegi verso i meccanismi di autenticazione supportati da Amazon DocumentDB.
- Differenze di privilegio: identifica i privilegi MongoDB senza equivalenti diretti di Amazon DocumentDB (ad esempio, ruoli di amministrazione dei cluster).

- Autenticazione delle applicazioni: aggiornamento delle stringhe di connessione e della gestione delle credenziali per le politiche sulle password di Amazon DocumentDB. Puoi utilizzare Secrets Manager per archiviare le tue credenziali e ruotare le password.
- Gestione degli account di servizio: stabilisci i processi per la gestione delle credenziali degli account di servizio in. AWS Secrets Manager
- Implementazione dei privilegi minimi: rivedi e perfeziona i controlli di accesso per implementare i principi dei privilegi minimi nel nuovo ambiente.

Encryption (Crittografia)

Garantisci che la crittografia a riposo e in transito sia in linea con i requisiti di conformità.

Configurazione della rete

Pianifica la configurazione del Virtual Private Cloud (VPC) e le regole del gruppo di sicurezza.

Considerazioni operative e di monitoraggio

Per mantenere l'affidabilità del sistema, l'individuazione dei carichi di lavoro dovrebbe includere anche:

- Strategia di backup e ripristino: valuta i metodi di backup esistenti e le funzionalità di backup di Amazon DocumentDB.
- AWS Backup integrazione: sfrutta AWS Backup la gestione centralizzata dei backup tra AWS servizi tra cui Amazon DocumentDB.
- CloudWatch metriche: mappa le metriche di monitoraggio MongoDB alle metriche di Amazon DocumentDB per CPU, memoria, CloudWatch connessioni e storage.
- Performance Insights: implementa Amazon DocumentDB Performance Insights per visualizzare il carico del database e analizzare i problemi di prestazioni con un'analisi dettagliata delle query.
- Profiler: configura il profiler Amazon DocumentDB per acquisire operazioni a esecuzione lenta (simile al profiler di MongoDB ma con impostazioni Amazon). DocumentDB-specific
  - Abilita tramite gruppi di parametri con soglie appropriate.
  - Analizza i dati del profiler per identificare le opportunità di ottimizzazione
- CloudWatch Eventi: configura il monitoraggio basato sugli eventi per gli eventi del cluster Amazon DocumentDB.
  - Configura le notifiche per eventi di backup, finestre di manutenzione e failover.

  - Effettua l'integrazione con Amazon SNS per avvisi e AWS Lambda risposte automatiche.
- Registrazione di audit: pianifica la configurazione della registrazione di audit per tenere traccia delle attività degli utenti e degli eventi rilevanti per la sicurezza.
- Monitoraggio avanzato: abilita il monitoraggio avanzato delle metriche granulari a intervalli di 1 secondo OS-level .

Crea un nuovo cluster di destinazione

Crea un cluster Amazon DocumentDB seguendo i passaggi indicati inCreazione di un cluster Amazon DocumentDB. Quando configuri il cluster per la migrazione, applica questi consigli:

- Classe di istanza: scegli R8G per ottenere prestazioni ottimali. Se R8G non è disponibile nella regione di destinazione, utilizza R6G. Scegli un'istanza quanto più grande possibile per ottenere il massimo rendimento a pieno carico. Riduci la scala una volta completata la migrazione.
- Versione del motore: più recente (8.0.1)
- Numero di istanze: scegli 1 istanza per ridurre al minimo i costi durante la migrazione. Scalabilità fino a 3 istanze per un'elevata disponibilità una volta completata la migrazione a pieno carico.
- Impostazioni di rete: assicurati che il tuo gruppo di sicurezza Amazon DocumentDB consenta le connessioni in entrata dal gruppo di sicurezza dell'istanza di replica DMS (per le migrazioni online) o dall'host di migrazione EC2 (per le migrazioni offline tramite mongorestore).

Migrazione dell'indice

La migrazione da MongoDB ad Amazon DocumentDB comporta il trasferimento non solo di dati ma anche di indici per mantenere le prestazioni delle query e ottimizzare le operazioni del database. Questa sezione illustra la procedura dettagliata e dettagliata per la migrazione degli indici da MongoDB ad Amazon DocumentDB garantendo al contempo compatibilità ed efficienza.

Utilizzo dello strumento di indicizzazione Amazon DocumentDB

Clona lo strumento di indicizzazione su GitHub

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

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

Esporta gli indici da MongoDB (se stai migrando da MongoDB)

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

Verifica gli indici

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

Indici di importazione

```
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'
```

Migrazione degli utenti

La migrazione degli utenti da MongoDB ad Amazon DocumentDB è essenziale per mantenere il controllo degli accessi, l'autenticazione e la sicurezza del database. Questa sezione descrive i passaggi dettagliati per migrare con successo gli utenti MongoDB preservandone i ruoli e le autorizzazioni utilizzando lo strumento di esportazione per gli utenti di Amazon DocumentDB.

Utilizzo dello strumento di esportazione degli utenti di Amazon DocumentDB

`Export Users tool`Esporta utenti e ruoli da MongoDB o Amazon DocumentDB in JavaScript file, che possono quindi essere utilizzati per ricrearli in un altro cluster.

Prerequisiti

```
# 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
```

Fase 1: Esportazione di utenti e ruoli

```
# 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"
```

Passaggio 2: modifica il file degli utenti

Le password non vengono esportate per motivi di sicurezza. Apri il `mongodb-users.js` file e aggiungi una password per ogni utente sostituendo il pwd valore in ogni `createUser` istruzione:

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

Passaggio 3: ripristino dei ruoli personalizzati in Amazon DocumentDB

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

Fase 4: ripristinare gli utenti su Amazon DocumentDB

```
# Import users after roles are created
mongosh \
  --tls \
  --host <target_endpoint>:27017 \
```

```
  --tlsCAFile global-bundle.pem \
  --username <username> \
  --password <password> \
  mongodb-users.js
```

Note importanti

- Le password non vengono esportate per motivi di sicurezza e devono essere aggiunte manualmente al file users.js.
- I ruoli devono essere importati prima degli utenti per garantire la corretta assegnazione dei ruoli.
- Lo strumento genera JavaScript file che possono essere eseguiti direttamente con la shell mongosh.
- I ruoli personalizzati e i relativi privilegi vengono preservati durante la migrazione.
- Questo approccio consente la revisione e la modifica delle autorizzazioni utente prima dell'importazione.

Questo metodo fornisce un approccio sicuro e flessibile alla migrazione di utenti e ruoli da MongoDB ad Amazon DocumentDB, consentendo al contempo la reimpostazione delle password durante il processo di migrazione.

Migrazione dei dati

Argomenti

- Migrazione online
- Migrazione offline

Migrazione online

Questa sezione fornisce passaggi dettagliati per eseguire una migrazione online da MongoDB ad Amazon DocumentDB utilizzando per AWS DMS abilitare tempi di inattività minimi e una replica continua. Per iniziare, configuri un cluster Amazon DocumentDB come destinazione e assicurati che l'istanza MongoDB sia configurata correttamente come origine, richiedendo in genere la modalità di set di repliche per l'acquisizione dei dati di modifica. Successivamente, crei un'istanza di replica DMS e definisci gli endpoint di origine e di destinazione con i dettagli di connessione necessari. Dopo aver convalidato gli endpoint, si configura e si avvia un'attività di migrazione che può includere il caricamento completo dei dati, la replica continua o entrambi.

Crea un nuovo cluster di destinazione

Consulta Crea un nuovo cluster di destinazione.

Configurare la fonte

MongoDB e Amazon DocumentDB possono entrambi fungere da fonti di migrazione, a seconda dello scenario:

- MongoDB come origine: comune durante la migrazione da un MongoDB locale o autogestito a un Amazon DocumentDB o altri servizi di database. AWS Richiede l'esecuzione in modalità set di repliche con un oplog di dimensioni adeguate (assicurati che sia dimensionato per contenere tutte le operazioni durante il pieno carico) per supportare l'acquisizione dei dati di modifica durante la migrazione.
- Amazon DocumentDB come sorgente: in genere utilizzato per la replica in più regioni, gli aggiornamenti di versione o la migrazione verso altri servizi di database come MongoDB Atlas. Richiede l'Abilitazione dei flussi di modificaimpostazione del `change_stream_log_retention_duration` parametro nel gruppo di parametri del cluster per acquisire le modifiche in corso durante la migrazione. Assicurati che le tue `change_stream_log_retention_duration` impostazioni siano sufficientemente grandi da coprire il tempo necessario per completare il caricamento completo.

Prima di iniziare la migrazione, configura la fonte per consentire AWS DMS l'accesso.

Crea un utente MongoDB con le autorizzazioni appropriate:

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

Configura rete e autenticazione.

Quando si configura la connettività di rete per la migrazione da MongoDB a DMS:

EC2-hosted Fonte MongoDB

- Modifica il gruppo di sicurezza EC2 per consentire il traffico in entrata dal gruppo di sicurezza dell'istanza di replica DMS.

- Aggiungi una regola per la porta TCP 27017 (o la tua porta MongoDB personalizzata).
- Utilizza l'ID del gruppo di sicurezza dell'istanza di replica DMS come fonte per un controllo preciso degli accessi.
- Assicurati che la sottorete dell'istanza EC2 abbia un percorso verso la sottorete dell'istanza di replica DMS.

On-premises Fonte MongoDB

- Configura il firewall per consentire le connessioni in entrata dagli indirizzi IP pubblici dell'istanza di replica DMS.
- Se utilizzi Direct Connect una VPN, assicurati che il routing sia corretto tra la rete e il VPC contenente l'istanza DMS.
- Verifica la connettività utilizzando i comandi telnet o nc dalla sottorete DMS al server MongoDB.

Fonte MongoDB Atlas

- Aggiungi gli indirizzi IP di un'istanza di replica DMS all'elenco degli indirizzi IP consentiti di MongoDB Atlas.
- Configura il peering VPC tra VPC e MongoDB Atlas AWS VPC se Atlas è in esecuzione. AWS
- Configura AWS PrivateLink la connettività privata (livello Enterprise), se in esecuzione su un altro provider cloud.
- Crea un utente dedicato con le read/write autorizzazioni appropriate.
- Usa una stringa di connessione MongoDB Atlas con la modalità SSL impostata su «verify-full».
- Assicurati che il log abbia una dimensione sufficiente per la durata della migrazione.

Fonte Amazon DocumentDB

Configura il gruppo di sicurezza Amazon DocumentDB di origine per consentire il traffico in entrata dal gruppo di sicurezza dell'istanza di replica DMS.

Crea un'istanza di replica DMS

Usa DMS Buddy on GitHub per creare un'infrastruttura di migrazione ottimale con impostazioni DMS e dimensioni delle istanze ottimali. Se preferisci configurare manualmente, segui questi passaggi:

1. Nella AWS DMS console, nel pannello di navigazione, scegli Migrate o replicate, quindi scegli Provisioned instances.
2. Inserisci i dettagli dell'istanza di replica:
   - Nome dell'istanza: scegli un nome univoco.
   - Classe di istanza: seleziona in base al carico di lavoro. Esempio: dms.r7i.large (piccoli carichi di lavoro), dms.r7i.4xlarge (carichi di lavoro di grandi dimensioni).
   - Versione del motore: 3.5.4
   - Memoria allocata: l'impostazione predefinita è 50 GB (aumentala se necessario). Ciò è determinato dalle dimensioni del documento updates/second e dalla durata del caricamento completo.
   - Multi-AZ Distribuzione: abilita per l'alta disponibilità, se necessario.
   - Scegli lo stesso VPC di Amazon DocumentDB.
   - Assicurati che i gruppi di sicurezza consentano il traffico in entrata dall'origine e da Amazon DocumentDB.
3. Scegli Crea istanza di replica e attendi che lo stato sia disponibile.

Crea endpoint DMS

Crea un endpoint di origine

Per una fonte MongoDB

1. Nella console DMS, nel pannello di navigazione, scegli Migrate or replicate, quindi scegli Endpoints.
2. Seleziona Crea endpoint.
3. Nella pagina Crea endpoint, scegli Endpoint di origine.
4. Nella sezione Configurazione dell'endpoint:
   - Inserisci un identificatore dell'endpoint univoco e significativo (ad esempio, «mongodb-source»).
   - Scegli MongoDB come motore di origine.
   - Per Accede al database di endpoint scegli Fornisci le informazioni di accesso manualmente.
   - Per il nome del server, inserisci il tuo. *MongoDB server DNS name/IP address*
   - Per Porta, inserisci 27017 (porta MongoDB predefinita).

- Per la modalità di autenticazione, scegli la modalità appropriata per la tua applicazione (password/SSL) (l'impostazione predefinita è Secrets Manager).
- Se la modalità di autenticazione è Password, fornisci:
  - Nome utente e password: inserisci le credenziali MongoDB.
  - Nome del database: il nome del database di origine.
  - Meccanismo di autenticazione: SCRAM-SHA-1 (predefinito) o meccanismo appropriato

5. Per la modalità Metadati, lasciate l'impostazione predefinita del documento.
6. Attributi di connessione aggiuntivi:

   - authSource=admin (se il database di autenticazione è diverso)
   - <your-replica-set-name>ReplicaSet= (richiesto per CDC)

Per una fonte Amazon DocumentDB

1. Nella console DMS, nel pannello di navigazione, scegli Migrate or replicate, quindi Endpoints.
2. Seleziona Crea endpoint.
3. Nella pagina Crea endpoint, scegli Source endpoint.
4. Nella sezione Configurazione dell'endpoint:

   - Inserisci un identificatore dell'endpoint univoco e significativo (ad esempio, «docdb-source»).
   - Scegli Amazon DocumentDB come motore di origine.
   - Per Accede al database di endpoint scegli Fornisci le informazioni di accesso manualmente.
   - Per il nome del server, inserisci il tuo*source Amazon DocumentDB cluster endpoint*.
   - Per Porta, inserisci 27017 (porta Amazon DocumentDB predefinita).
   - Per la modalità SSL, scegli verify-full (consigliato per Amazon DocumentDB).
   - Per il certificato CA, scegli il certificato CA principale di Amazon RDS.
   - Per la modalità di autenticazione, scegli la modalità appropriata per la tua applicazione (password/SSL) (l'impostazione predefinita è Secrets Manager).
   - Se la modalità di autenticazione è Password, fornisci:
     - Nome utente e password: inserisci le credenziali di Amazon DocumentDB.
     - Nome del database: il nome del database di origine.
     - Meccanismo di autenticazione: SCRAM-SHA-1 (predefinito) o meccanismo appropriato

5. Per la modalità Metadati, lasciate l'impostazione predefinita del documento.

Crea un endpoint di destinazione (Amazon DocumentDB)

1. Nella console DMS, nel pannello di navigazione, scegli Migrate or replicate, quindi Endpoints.
2. Seleziona Crea endpoint.
3. Nella pagina Crea endpoint, scegli Target endpoint.
4. Nella sezione di configurazione dell'endpoint:

   - Inserisci un identificatore dell'endpoint univoco e significativo (ad esempio, «docdb-target»).
   - Scegli Amazon DocumentDB come motore di Target.
   - Per Access to endpoint database, scegli il metodo che desideri utilizzare per autenticare l'accesso al database:
     - Se scegli AWS Secrets Manager, scegli il segreto in cui archiviare le tue credenziali Amazon DocumentDB nel campo Segreto.
     - Se scegli Fornisci le informazioni di accesso manualmente:
       - Per Nome server, inserisci il tuo*target Amazon DocumentDB cluster endpoint*.
       - Per Porta, inserisci 27017 (porta Amazon DocumentDB predefinita).
       - Per la modalità SSL, scegli verify-full (consigliato per Amazon DocumentDB).
       - Per CA Certificate, scarica e specifica il pacchetto di certificati CA per la verifica SSL.
       - Per la modalità di autenticazione, scegli la modalità appropriata per la tua applicazione (password/SSL) (l'impostazione predefinita è Secrets Manager).
       - Se la modalità di autenticazione è Password, fornisci:
         - Nome utente e password: inserisci le credenziali di Amazon DocumentDB.
         - Nome del database: il nome del database di origine.
         - Meccanismo di autenticazione: SCRAM-SHA-1 (predefinito) o meccanismo appropriato
5. Per la modalità Metadati, lasciate l'impostazione predefinita del documento.

Crea attività di replica

1. Nella console DMS, nel pannello di navigazione, scegli Migra o replica, quindi scegli Attività.
2. Scegli Create task (Crea attività).
3. Nella pagina Crea attività, nella sezione Configurazione dell'attività:

   - Inserisci un identificatore di attività univoco e significativo (ad esempio, «mongodb-docdb-replication»).

- Scegli l'endpoint di origine che hai creato in precedenza nel menu a discesa Endpoint del database di origine.
- Scegli l'endpoint di destinazione che hai creato in precedenza nel menu a discesa Endpoint del database di Target.
- Scegli l'istanza di replica creata in precedenza nel menu a discesa Provisioned instance.
- Per Tipo di attività, scegli Migra e replica.
- Per quanto tempo intendi continuare a replicare dall'origine alla destinazione? , scegli Per un periodo limitato.

4. Nella sezione Impostazioni:

   - Per la modalità di preparazione della tabella Target, scegli Non fare nulla. Altrimenti, le raccolte e gli indici già creati verranno eliminati.
   - Per i registri delle attività, seleziona la casella Attiva i CloudWatch registri.
   - Lascia i valori predefiniti per tutte le altre impostazioni.

5. Torna nella parte superiore della sezione Impostazioni, per la modalità di modifica, scegli l'editor JSON e imposta i seguenti attributi:

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

   `MaxFullLoadSubTasks`Impostato in base al numero di raccolte e alla capacità dell'istanza. Usa valori più bassi (2—4) per raccolte più piccole o più piccole e valori più alti (8—16) per molte raccolte di grandi dimensioni. Per il dimensionamento automatico, vedi DMS Buddy su. https://github.com/awslabs/amazon-documentdb-tools/tree/master/migration/dms_buddy GitHub

6. Nella sezione Mappature delle tabelle, aggiungi una nuova regola di selezione:

   - Per il nome dello schema, aggiungi il database di origine da migrare. Usa% per specificare più database.
   - Per il nome della tabella Schema, aggiungi la raccolta di sorgenti da migrare. Usa% per specificare più raccolte.

- Per Azione, lascia l'impostazione predefinita di Includi

7. Per raccolte di grandi dimensioni (oltre 100 GB), aggiungi la regola delle impostazioni della tabella:

   - Per il nome dello schema, aggiungi il database di origine da migrare. Usa% per specificare più database.
   - Per il nome della tabella Schema, aggiungi la raccolta di sorgenti da migrare. Usa% per specificare più raccolte.
   - Per Numero di partizioni, impostato in base al carico di lavoro. Per un numero limitato di raccolte di grandi dimensioni, imposta le partizioni ≥ `MaxFullLoadSubTasks` per tenere occupati tutti i lavoratori. Per molte raccolte che vengono caricate in parallelo, impostate le partizioni ≤ `MaxFullLoadSubTasks` poiché il numero di raccolte stesso fornisce un parallelismo sufficiente.

8. Nella sezione Valutazione della premigrazione, assicurati che sia disattivata. La valutazione della premigrazione non è supportata per Amazon DocumentDB come destinazione e deve essere disabilitata.

Migrazione offline

Questa sezione descrive il processo per eseguire una migrazione offline da un'istanza MongoDB autogestita ad Amazon DocumentDB utilizzando strumenti MongoDB nativi: e. `mongodump` `mongorestore`

Prerequisiti

Requisiti MongoDB di origine

- Accesso all'istanza MongoDB di origine con le autorizzazioni appropriate.
- Install`mongodump`. se necessario (viene installato durante l'installazione di MongoDB).
- Assicurati che ci sia abbastanza spazio su disco per i file di dump.

Requisiti di Amazon DocumentDB target

- Assicurati di aver effettuato il provisioning di un cluster Amazon DocumentDB.
- Assicurati che sia presente un'istanza EC2 nello stesso VPC di Amazon DocumentDB per facilitare la migrazione.

- La connettività di rete deve essere disponibile tra l'ambiente di origine e Amazon DocumentDB.
- mongorestore deve essere installato sull'istanza EC2 di migrazione.
- Le autorizzazioni IAM appropriate devono essere configurate per accedere ad Amazon DocumentDB,

Requisiti generali

- AWS CLI devono essere configurati (se si utilizzano AWS servizi per lo storage intermedio)
- Deve essere disponibile una larghezza di banda sufficiente per il trasferimento dei dati.
- La finestra di inattività deve essere approvata (se si esegue una migrazione in tempo reale, si consiglia di prendere in considerazione altri approcci)

Prepara un cluster Amazon DocumentDB

Consulta Crea un nuovo cluster di destinazione.

Esegui il dump dei dati (mongodump)

Scegli una delle seguenti opzioni per creare un file di dump:

- Opzione 1: Basic

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

- Opzione 2: migliore controllo e prestazioni

  Comprime l'output con gzip, scarica le raccolte in parallelo, abilita TLS e legge da una replica secondaria, se disponibile:

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

- Opzione 3: database di grandi dimensioni

  Scarica un database e una raccolta specifici, filtra i documenti per interrogazione e li restituisce in un unico archivio compresso:

```
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
```

Trasferisci i file di dump nell'ambiente di ripristino

Scegliete un metodo appropriato in base alle dimensioni del dump:

- Piccolo: copia direttamente sul tuo computer di migrazione (istanza EC2 creata in precedenza):

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

- Medio: usa Amazon S3 come storage intermedio:

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

- Grande: per database molto grandi, considera un AWS DataSync trasferimento fisico.

Ripristina i dati su Amazon DocumentDB (mongorestore)

Prima di iniziare il processo di ripristino, crea gli indici in Amazon DocumentDB. Puoi utilizzare lo strumento Amazon DocumentDB Index per esportare e importare indici.

Scegli una delle seguenti opzioni per ripristinare i dati:

- Opzione 1: ripristino di base

```
mongorestore \
```

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

- Opzione 2: controllo e prestazioni migliori

  Ripristina i dump compressi con gzip con raccolta parallela e inserimento di documenti. Salta il ripristino dell'indice poiché gli indici sono pre-creati:

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

- Opzione 3: database di grandi dimensioni o controlli specifici

  Ripristina da un archivio compresso con filtro dello spazio dei nomi per includere o escludere database e raccolte specifici. Salta il ripristino dell'indice poiché gli indici sono precreati:

```
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
```

Monitoraggio

Questa sezione fornisce un processo di monitoraggio dettagliato per tenere traccia dell'avanzamento, delle prestazioni e dello stato di una migrazione in corso.

Le fasi di monitoraggio si applicano indipendentemente dal metodo di migrazione (AWS DMS mongodump/mongorestore,, o altri strumenti).

AWS DMS Monitoraggio della migrazione (se applicabile)

Monitora le seguenti CloudWatch metriche chiave:

Metriche relative alla fase di carico completo

- FullLoadThroughputBandwidthTarget— Larghezza di banda di rete (KB/second) a pieno carico
- FullLoadThroughputRowsTarget— Numero di rows/documents caricamenti al secondo
- FullLoadThroughputTablesTarget— Numero di tables/collections completamenti al minuto
- FullLoadProgressPercent— Percentuale di carico completo completato
- TablesLoaded— Numero di caricamenti tables/collections effettuati con successo
- TablesLoading— Numero di caricamenti tables/collections attualmente in corso
- TablesQueued— Numero di tables/collections attese da caricare
- TablesErrored— Numero di quelli tables/collections che non sono stati caricati

Metriche di fase CDC

- CDCLatencyTarget— Ritardo (secondi) tra il cambio di origine e l'applicazione di destinazione
- CDCLatencySource— Ritardo (secondi) tra la modifica della sorgente e la lettura da parte del DMS
- CDCThroughputRowsTarget— Righe al secondo applicate durante la replica in corso
- CDCThroughputBandwidthTarget— Larghezza di banda di rete (KB/second) durante il CDC
- CDCIncomingChanges— Numero di eventi di modifica ricevuti dalla fonte
- CDCChangesMemoryTarget— Memoria utilizzata (MB) per memorizzare le modifiche sul lato di destinazione

Metriche relative alle risorse

- Utilizzo della CPU: utilizzo della CPU dell'istanza di replica

- FreeableMemory— Memoria disponibile sull'istanza di replica
- FreeStorageSpace— Storage disponibile sull'istanza di replica
- NetworkTransmitThroughput— Throughput di rete per l'istanza di replica
- NetworkReceiveThroughput— Throughput di rete per l'istanza di replica

Metriche di errore

- ErrorsCount— Numero totale di errori durante la migrazione
- TableErrorsCount— Numero di errori specifici della tabella
- RecordsErrorsCount— Numero di errori specifici dei record

Crea CloudWatch allarmi per metriche critiche, `CPUUtilization` ad esempio `CDCLatencyTarget` per ricevere notifiche se le prestazioni della migrazione peggiorano.

Log DMS (log) CloudWatch

1. Vai alla console Amazon CloudWatch Logs.
2. Trova e scegli nel tuo gruppo di log. Sarà simile a «dms-tasks —».
3. Cerca i flussi di log che potrebbero contenere informazioni sugli errori:
   - Stream con «error» nel nome
   - Stream con ID di attività o nomi di endpoint
   - I flussi di log più recenti durante il periodo della migrazione
4. All'interno di questi stream, cerca parole chiave come:
   - «errore»
   - «eccezione»
   - «fallito»
   - «avviso»

stato dell'attività DMS (utilizzando AWS CLI)

```
aws dms describe-replication-tasks \
  --filters Name=replication-task-id,Values=<task_id> \
```

```
  --query "ReplicationTasks[0].Status"
```

Flusso di stato previsto:

creazione → pronto → in esecuzione → arresto → interrotto (o fallito)

Monitora usando **`docdb-dashboarder`**

Lo `docdb-dashboarder` strumento fornisce un monitoraggio completo per i cluster Amazon DocumentDB generando automaticamente dashboard con metriche prestazionali essenziali. CloudWatch Queste dashboard mostrano metriche critiche a livello di cluster (ritardo di replica, contatori delle operazioni), metriche a livello di istanza (CPU, memoria, connessioni) e metriche di archiviazione (utilizzo del volume, archiviazione di backup). Per gli scenari di migrazione, lo strumento offre dashboard specializzate che tracciano l'avanzamento della migrazione con metriche come il ritardo di replica CDC e i tassi operativi. Le dashboard possono monitorare più cluster contemporaneamente e includere il supporto per le istanze. NVMe-backed Visualizzando queste metriche, i team possono identificare in modo proattivo i colli di bottiglia nelle prestazioni, ottimizzare l'allocazione delle risorse e garantire il corretto funzionamento delle distribuzioni di Amazon DocumentDB. Lo strumento elimina la necessità della creazione manuale di dashboard fornendo al contempo un monitoraggio coerente in tutti gli ambienti. Per istruzioni di configurazione e opzioni di configurazione avanzate, consulta il repository Amazon DocumentDB Dashboarder Tool. GitHub

Convalida

Argomenti

- Lista di controllo per la convalida
- Convalida dello schema e dell'indice
- Campionamento dei dati e convalida a livello di campo
- Validazione tramite strumento DataDiffer

Questa sezione fornisce un processo di convalida dettagliato per garantire la coerenza, l'integrità e la compatibilità delle applicazioni dopo la migrazione.

Le fasi di convalida si applicano indipendentemente dal metodo di migrazione (AWS DMS, mongodump/mongorestore, o altri strumenti).

Lista di controllo per la convalida

Verifica che il numero di documenti in ogni raccolta corrisponda tra origine e destinazione:

Fonte MongoDB

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

Obiettivo Amazon DocumentDB

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

Convalida dello schema e dell'indice

Verifica che:

- tutte le raccolte esistono nella destinazione.
- gli indici vengono replicati correttamente.
- le definizioni dello schema (se applicate) sono identiche.

Controlla le raccolte (origine e destinazione)

```
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();"
```

Controlla gli indici (origine e destinazione)

```
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()"
```

Confronta l'elenco delle raccolte per assicurarti che non vi siano raccolte mancanti o aggiuntive.

Verifica gli indici controllando i nomi degli indici, le definizioni delle chiavi, i vincoli univoci e gli indici TTL (se presenti).

Controlla le regole di convalida dello schema (se usi la convalida dello schema in MongoDB)

```
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"
```

Campionamento dei dati e convalida a livello di campo

È possibile campionare i documenti in modo casuale e confrontare i campi tra origine e destinazione.

Campionamento manuale

Recupera cinque documenti a caso (fonte):

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

Recupera gli stessi ID del documento (destinazione):

```
mongosh --host <target_endpoint> \
  --port 27017 \
  --username <username> \
  --password <password> \
  --tls --tlsCAFile global-bundle.pem \
  --eval "db.<collection>.find({ _id: { \$in: [<list_of_ids>] } })"
```

Campionamento automatico

```
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']}")
```

Validazione tramite strumento DataDiffer

Lo DataDiffer strumento attivo GitHub fornisce un modo affidabile per confrontare i dati tra database di origine e di destinazione.

Prerequisiti

I seguenti prerequisiti devono essere soddisfatti prima di installare lo DataDiffer strumento:

- Python3.7+
- PyMongo libreria
- Connettività di rete ai cluster Amazon DocumentDB di origine e di destinazione

Configurazione e installazione

Clona il repository e accedi alla directory DataDiffer

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

Installa le dipendenze richieste

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

Convalida dei dati in esecuzione

Crea un file di configurazione (ad esempio, config.json) con i dettagli della connessione

```
{
  "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
  }
```

```
}
```

Esegui lo strumento DataDiffer

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

Per raccolte di grandi dimensioni, utilizza il campionamento per convalidare un sottoinsieme di dati

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

Per convalidare più raccolte, create file di configurazione separati o utilizzate la modalità batch

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

Interpretazione dei risultati

Lo strumento produrrà:

- Documenti totali in origine e destinazione
- Numero di documenti corrispondenti
- Numero di documenti mancanti
- Numero di documenti con differenze
- Rapporto dettagliato delle differenze (se presenti)

Best practice

Di seguito sono riportate le migliori pratiche per l'utilizzo dello DataDiffer strumento:

- Esegui per fasi: prima convalida il numero di documenti, poi campiona i documenti chiave e infine esegui un confronto completo, se necessario.
- Verifica le differenze di schema: Amazon DocumentDB presenta alcune limitazioni rispetto a MongoDB. Lo strumento evidenzierà tipi o strutture di dati incompatibili.
- Convalida durante i periodi di silenzio: esegui la convalida quando le operazioni di scrittura sono minime per garantire la coerenza.
- Monitora l'utilizzo delle risorse: il processo di confronto può richiedere molte risorse. Regola le dimensioni del batch e il numero di thread di conseguenza.

- Convalida degli indici: dopo la convalida dei dati, assicurati che tutti gli indici richiesti siano stati creati nel cluster Amazon DocumentDB di destinazione.
- Risultati della convalida dei documenti: tieni un registro dei risultati di convalida per ogni raccolta come parte della documentazione di migrazione.

Migrazione da Couchbase Server

Argomenti

- Introduzione
- Confronto con Amazon DocumentDB
- Individuazione
- Pianificazione
- Migrazione
- Convalida

Introduzione

Questa guida presenta i punti chiave da considerare durante la migrazione da Couchbase Server ad Amazon DocumentDB. Spiega le considerazioni per le fasi di scoperta, pianificazione, esecuzione e convalida della migrazione. Spiega inoltre come eseguire migrazioni offline e online.

Confronto con Amazon DocumentDB

	Server Couchbase	Amazon DocumentDB
Organizzazione dei dati	Nelle versioni 7.0 e successive, i dati sono organizzati in bucket, ambiti e raccolte. Nelle versioni precedenti, i dati sono organizzati in bucket.	I dati sono organizzati in database e raccolte.
Compatibilità	Esistono API separate per ogni servizio (ad esempio dati, indice, ricerca, ecc.). Le ricerche secondarie utilizzano SQL++	Amazon DocumentDB è compatibile con l'API MongoDB. compatibility.html

	Server Couchbase	Amazon DocumentDB
	(precedentemente noto come N1QL), un linguaggio di interroga zione basato su ANSI-standard SQL, quindi familiare a molti sviluppatori.	
Architettura	Lo storage è collegato a ciascuna istanza del cluster. Non è possibile scalare l'elaborazione indipende ntemente dallo storage.	Amazon DocumentDB è progettat o per il cloud e per evitare i limiti delle architetture di database tradizionali. I livelli di elaborazi one e storage sono separati in Amazon DocumentDB e il livello di elaborazione può essere scalato indipendentemente dallo storage. how-it-works.html
Aggiungi capacità di lettura su richiesta	I cluster possono essere scalati aggiungendo istanze. Poiché lo storage è collegato all'istanza in cui il servizio è in esecuzione, il tempo necessario per la scalabilità orizzontale dipende dalla quantità di dati che devono essere spostati nella nuova istanza o ribilanciati.	Puoi ottenere la scalabilità in lettura per il tuo cluster Amazon DocumentDB creando fino a 15 repliche Amazon DocumentD B nel cluster. Non vi è alcun impatto sul livello di storage.
Ripristina rapidamente in caso di guasto del nodo	I cluster dispongono di funzionalità di failover automatico, ma il tempo necessario per ripristinare la piena funzionalità del cluster dipende dalla quantità di dati da spostare nella nuova istanza.	Amazon DocumentDB è in grado di eseguire il failover del sistema primario in genere entro 30 secondi e ripristinare la piena funzionalità del cluster in 8-10 minuti, indipendentemente dalla quantità di dati nel cluster.
Scala lo storage man mano che i dati crescono	Per i cluster autogestiti, lo storage e iOS non sono scalabili automatic amente.	Lo storage Amazon DocumentD B e iOS si scalano automatic amente.

	Server Couchbase	Amazon DocumentDB
Esegui il backup dei dati senza influire sulle prestazioni	I backup vengono eseguiti dal servizio di backup e non sono abilitati per impostazione predefinita. Poiché lo storage e l'elaborazione non sono separati, può esserci un impatto sulle prestazioni.	I backup di Amazon DocumentDB sono abilitati per impostazione predefinita e non possono essere disattivati. I backup sono gestiti dal livello di storage, quindi non hanno alcun impatto sul livello di elaborazione. Amazon DocumentDB supporta il ripristino da un'istantanea del cluster e il ripristino in un determinato momento. backup_restore-point_in_time_recovery.html
Durabilità dei dati	In un cluster possono esserci un massimo di 3 copie di replica dei dati per un totale di 4 copie. Ogni istanza in cui il servizio dati è in esecuzione avrà 1, 2 o 3 copie di replica attive dei dati.	Amazon DocumentDB conserva 6 copie dei dati indipendentemente dal numero di istanze di calcolo presenti, con un quorum di scrittura di 4 e persistent true. I clienti ricevono una conferma dopo che il livello di storage ha mantenuto 4 copie dei dati.
Coerenza	È supportata la coerenza immediata K/V delle operazioni. L'SDK di Couchbase indirizza K/V le richieste all'istanza specifica che contiene la copia attiva dei dati, quindi una volta riconosciuto l'aggiornamento, il cliente ha la garanzia di leggerlo. La replica degli aggiornamenti ad altri servizi (indice, ricerca, analisi, gestione degli eventi) è infine coerente.	Le repliche di Amazon DocumentDB alla fine sono coerenti. Se sono richieste letture coerenti immediate, il cliente può leggere dall'istanza principale.

	Server Couchbase	Amazon DocumentDB
Replica	Cross-Data Center Replication (XDCR) fornisce una replica filtrata e attiva dei dati in passive/active molte topologie.	I cluster globali di Amazon DocumentDB forniscono la replica attiva-passiva in topologie 1:many (fino a 10).

Individuazione

La migrazione ad Amazon DocumentDB richiede una conoscenza approfondita del carico di lavoro del database esistente. Il rilevamento del carico di lavoro è il processo di analisi della configurazione e delle caratteristiche operative del cluster Couchbase (set di dati, indici e carico di lavoro) per garantire una transizione senza interruzioni con interruzioni minime.

Configurazione del cluster

Couchbase utilizza un'architettura incentrata sui servizi in cui ogni funzionalità corrisponde a un servizio. Esegui il seguente comando sul tuo cluster Couchbase per determinare quali servizi vengono utilizzati (vedi Ottenere informazioni sui nodi): https://docs.couchbase.com/server/current/rest-api/rest-node-get-info.html

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

Output di esempio:

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

I servizi Couchbase includono quanto segue:

Servizio dati (kv)

Il servizio dati fornisce read/write l'accesso ai dati in memoria e su disco.

Amazon DocumentDB supporta K/V le operazioni sui dati JSON tramite l'API MongoDB.

Servizio di interrogazione (n1ql)

Il servizio di interrogazione supporta l'interrogazione di dati JSON tramite SQL++.

Amazon DocumentDB supporta l'interrogazione di dati JSON tramite l'API MongoDB.

Servizio di indicizzazione (indice)

Il servizio di indicizzazione crea e gestisce indici sui dati, consentendo interrogazioni più rapide.

Amazon DocumentDB supporta un indice primario predefinito e la creazione di indici secondari su dati JSON tramite l'API MongoDB.

Servizio di ricerca (fts)

Il servizio di ricerca supporta la creazione di indici per la ricerca a testo completo.

La funzionalità di ricerca full-text nativa di Amazon DocumentDB consente di eseguire query di testo di base su set di dati testuali di grandi dimensioni utilizzando indici di testo per scopi speciali utilizzando l'API. MongoDB Per funzionalità avanzate come la ricerca fuzzy, la ricerca multilingue e la ricerca tra raccolte, utilizza l'integrazione Zero-ETL di Amazon DocumentDB con Amazon Service. OpenSearch

Servizio di analisi (cbas)

Il servizio di analisi supporta l'analisi dei dati JSON quasi in tempo reale.

Amazon DocumentDB supporta query ad hoc sui dati JSON tramite l'API MongoDB. Puoi anche eseguire query complesse sui tuoi dati JSON in Amazon DocumentDB utilizzando Apache Spark in esecuzione su Amazon EMR.

Servizio Eventing (eventing)

Il servizio di eventing esegue la logica aziendale definita dall'utente in risposta alle modifiche dei dati.

Amazon DocumentDB automatizza i carichi di lavoro basati sugli eventi richiamando AWS Lambda funzioni ogni volta che i dati cambiano con il tuo cluster Amazon DocumentDB.

Servizio di backup (backup)

Il servizio di backup pianifica backup completi e incrementali dei dati e l'unione dei backup di dati precedenti.

Amazon DocumentDB esegue continuamente il backup dei dati su Amazon S3 con un periodo di conservazione di 1—35 giorni, in modo da poterli ripristinare rapidamente in qualsiasi momento entro il periodo di conservazione del backup. Amazon DocumentDB acquisisce anche istantanee automatiche dei tuoi dati come parte di questo processo di backup continuo. Puoi anche gestire il backup e il ripristino di Amazon DocumentDB con. AWS Backup.

Caratteristiche operative

Utilizza il Discovery Tool for Couchbase per ottenere le seguenti informazioni sul set di dati, sugli indici e sul carico di lavoro. Queste informazioni ti aiuteranno a dimensionare il tuo cluster Amazon DocumentDB.

Set di dati

Lo strumento recupera le seguenti informazioni sul bucket, sull'ambito e sulla raccolta:

1. nome bucket
2. Tipo di bucket
3. nome dell'ambito
4. nome della collezione
5. dimensione totale (byte)
6. articoli totali
7. dimensione dell'articolo (byte)

Indici

Lo strumento recupera le seguenti statistiche sugli indici e tutte le definizioni degli indici per tutti i bucket. Tieni presente che gli indici primari sono esclusi poiché Amazon DocumentDB crea automaticamente un indice primario per ogni raccolta.

1. nome bucket
2. nome dell'ambito
3. nome della collezione

4. nome dell'indice
5. dimensione dell'indice (byte)

Carico di lavoro

Lo strumento recupera le metriche delle K/V query N1QL. K/V i valori delle metriche vengono raccolti a livello di bucket e le metriche SQL++ vengono raccolte a livello di cluster.

Le opzioni della riga di comando dello strumento sono le seguenti:

```
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>
```

Di seguito è illustrato un esempio di comando:

```
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 i valori delle metriche si baseranno su campioni ogni 10 minuti nell'ultima settimana (vedi metodo HTTP e URI). I valori delle metriche SQL++ si baseranno su campioni ogni 1 secondo negli ultimi 60 secondi (vedi Etichette generali). L'output del comando sarà contenuto nei seguenti file:

collection-stats.csv: bucket, ambito e informazioni sulla raccolta

```
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 — nomi e dimensioni degli indici

```
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: ottieni, imposta ed elimina le metriche per tutti i bucket

```
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++ seleziona, elimina e inserisce le metriche per il cluster

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

indexes- <bucket-name>.txt — definizioni degli indici di tutti gli indici nel bucket. Tieni presente che gli indici primari sono esclusi poiché Amazon DocumentDB crea automaticamente un indice primario per ogni raccolta.

```
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`)
...
```

Pianificazione

Nella fase di pianificazione determinerai i requisiti del cluster Amazon DocumentDB e la mappatura dei bucket, degli ambiti e delle raccolte Couchbase nei database e nelle raccolte Amazon DocumentDB.

Requisiti del cluster Amazon DocumentDB

Usa i dati raccolti nella fase di scoperta per dimensionare il tuo cluster Amazon DocumentDB. Consulta Instance sizing per ulteriori informazioni sul dimensionamento del cluster Amazon DocumentDB.

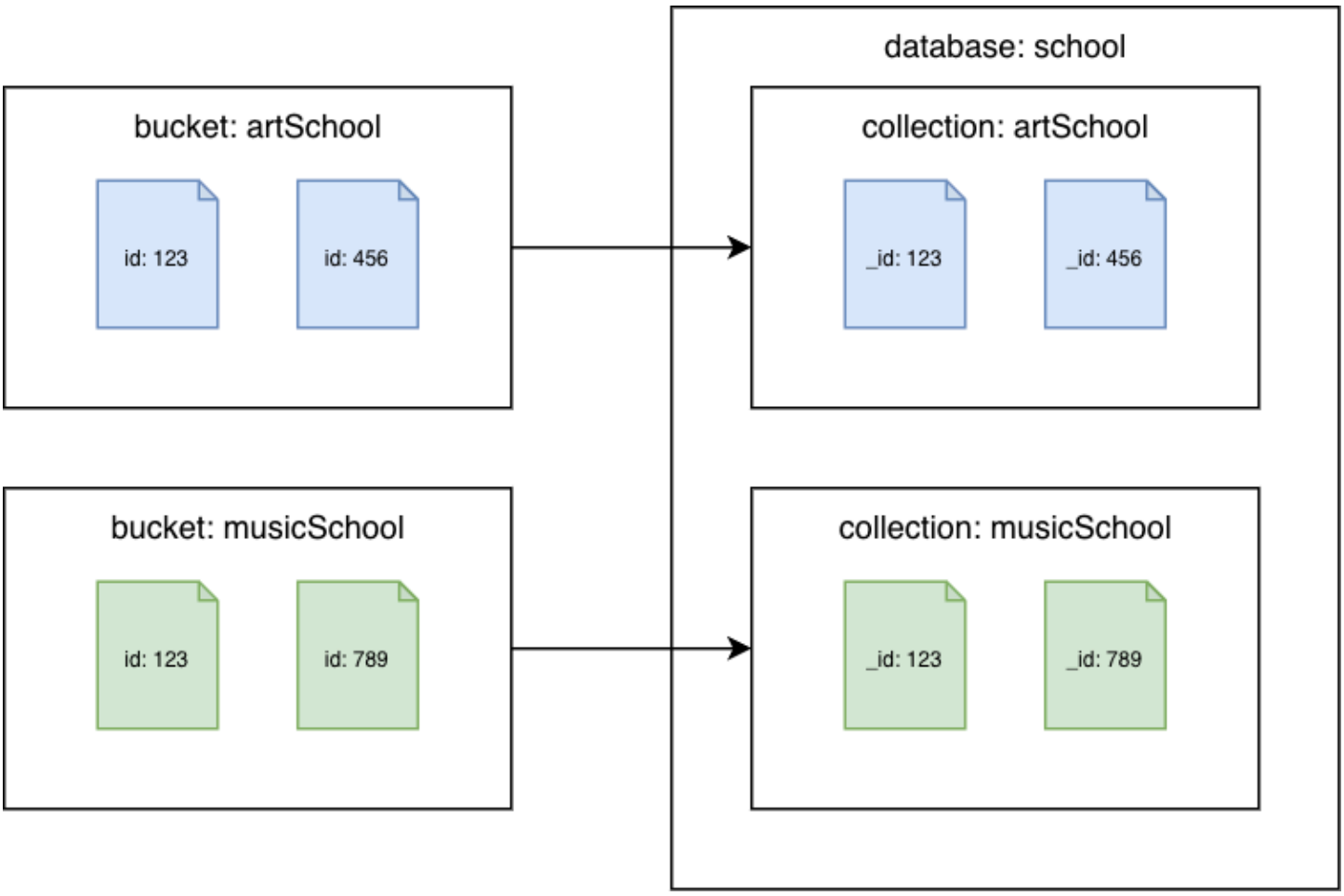
Mappatura di bucket, ambiti e raccolte su database e raccolte

Determina i database e le raccolte che esisteranno nei tuoi cluster Amazon DocumentDB. Considera le seguenti opzioni a seconda di come sono organizzati i dati nel tuo cluster Couchbase. Queste non sono le uniche opzioni, ma forniscono punti di partenza da prendere in considerazione.

Couchbase Server 6.x o versioni precedenti

I bucket Couchbase si adattano alle raccolte Amazon DocumentDB

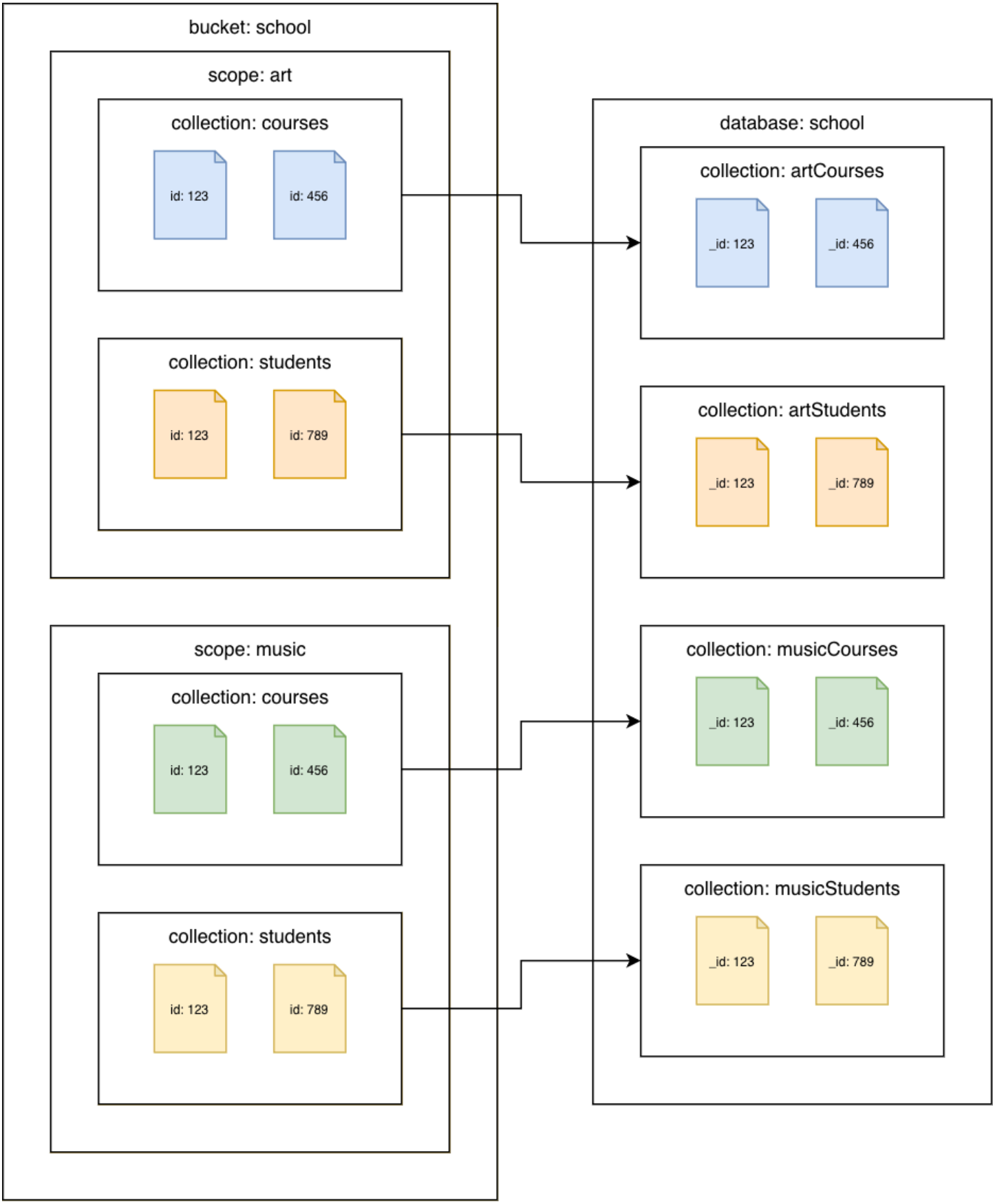
Esegui la migrazione di ogni bucket in una raccolta Amazon DocumentDB diversa. In questo scenario, il valore del documento Couchbase verrà utilizzato come `id` valore Amazon DocumentDB. `_id`



Couchbase Server 7.0 o successivo

Da raccolte Couchbase a raccolte Amazon DocumentDB

Esegui la migrazione di ogni raccolta in una raccolta Amazon DocumentDB diversa. In questo scenario, il valore del documento Couchbase verrà utilizzato come `id` valore Amazon DocumentDB. `_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

Migrazione

Migrazione dell'indice

La migrazione ad Amazon DocumentDB comporta il trasferimento non solo di dati ma anche di indici per mantenere le prestazioni delle query e ottimizzare le operazioni del database. Questa sezione illustra la procedura dettagliata e dettagliata per la migrazione degli indici su Amazon DocumentDB garantendo al contempo compatibilità ed efficienza.

Usa Amazon Q (disponibile nella Console di gestione AWS o nel tuo IDE) per convertire le istruzioni SQL++ in comandi Amazon DocumentDB. `CREATE INDEX` `createIndex()`

1. Carica i <bucket name>file indexes-.txt creati dal Discovery Tool per Couchbase.
2. Inserisci il seguente prompt:

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

Amazon Q genererà `createIndex()` comandi Amazon DocumentDB equivalenti. Tieni presente che potresti dover aggiornare i nomi delle raccolte in base a come hai mappato i bucket, gli ambiti e le raccolte Couchbase alle raccolte Amazon DocumentDB.

Ad esempio:

indexes-beer-sample.txt

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

Esempio di output Amazon Q (estratto):

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

db.beerSample.createIndex(
  { "code": 1 },
```

```
  {
    "name": "code",
    "background": true,
    "partialFilterExpression": { "type": "brewery" }
  }
)
```

Per tutti gli indici che Amazon Q non è in grado di convertire, consulta Gestione degli indici e degli indici di Amazon DocumentDB e delle proprietà degli indici per ulteriori informazioni.

Codice di refactoring per utilizzare le API MongoDB

I client utilizzano gli SDK Couchbase per connettersi a Couchbase Server. I client Amazon DocumentDB utilizzano i driver MongoDB per connettersi ad Amazon DocumentDB. Tutte le lingue supportate dagli SDK Couchbase sono supportate anche dai driver MongoDB. Consulta MongoDB Drivers per ulteriori informazioni sul driver per la tua lingua.

Poiché le API sono diverse tra Couchbase Server e Amazon DocumentDB, dovrai rifattorizzare il codice per utilizzare le API MongoDB appropriate. Puoi utilizzare Amazon Q per convertire le chiamate K/V API e le query SQL++ nelle API MongoDB equivalenti:

1. Carica i file del codice sorgente.
2. Inserisci il seguente prompt:

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

Utilizzando l'esempio di codice Python di Hello Couchbase, Amazon Q genera quanto segue (estratto):

```
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")
```

Fai riferimento a Connessione programmatica ad Amazon DocumentDB per esempi di connessione ad Amazon DocumentDB in Python, PHP, Go, Java, Node.js C#/.NET, R e Ruby.

Seleziona l'approccio alla migrazione

Quando si esegue la migrazione dei dati su Amazon DocumentDB, sono disponibili due opzioni:

1. migrazione offline
2. migrazione online

> **Note**
>
> Amazon DocumentDB utilizza il `_id` campo come chiave primaria per ogni documento. Gli approcci di migrazione descritti in questa guida memorizzano la chiave del documento Couchbase nel campo. `_id` Per la migrazione offline, questa operazione viene effettuata tramite l'`--include-key _id`opzione in. `cbexport` Per la migrazione online, questa operazione viene effettuata tramite il connettore Kafka. `ProvidedInKeyStrategy`

Migrazione offline

Prendi in considerazione una migrazione offline quando:

- I tempi di inattività sono accettabili: la migrazione offline comporta l'interruzione delle operazioni di scrittura nel database di origine, l'esportazione dei dati e quindi l'importazione in Amazon DocumentDB. Questo processo comporta tempi di inattività per l'applicazione. Se l'applicazione o il carico di lavoro sono in grado di tollerare questo periodo di indisponibilità, la migrazione offline è un'opzione praticabile.

- Migrazione di set di dati più piccoli o realizzazione di prove concettuali: per set di dati più piccoli, il tempo necessario per il processo di esportazione e importazione è relativamente breve, il che rende la migrazione offline un metodo rapido e semplice. È inoltre adatto per ambienti di sviluppo, test e proof-of-concept in cui i tempi di inattività sono meno critici.
- La semplicità è una priorità: il metodo offline, che utilizza cbexport e mongoimport, è generalmente l'approccio più semplice per migrare i dati. Evita le complessità dell'acquisizione dei dati di modifica (CDC) coinvolta nei metodi di migrazione online.
- Non è necessario replicare le modifiche continue: se il database di origine non riceve attivamente le modifiche durante la migrazione o se tali modifiche non sono fondamentali da acquisire e applicare alla destinazione durante il processo di migrazione, è appropriato un approccio offline.

Argomenti

- Couchbase Server 6.x o versioni precedenti
- Couchbase Server 7.0 o successivo

Couchbase Server 6.x o versioni precedenti

Dal bucket Couchbase alla raccolta Amazon DocumentDB

Esporta i dati utilizzando cbexport json per creare un dump JSON di tutti i dati nel bucket. Per l'`--format`opzione puoi usare o. `lines` `list`

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

Importa i dati in una raccolta Amazon DocumentDB utilizzando mongoimport con l'opzione appropriata per importare le righe o l'elenco:

linee:

```
mongoimport \
  --db <database> \
```

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

elenco:

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

Couchbase Server 7.0 o successivo

Per eseguire una migrazione offline, usa gli strumenti cbexport e mongoimport:

Bucket Couchbase con ambito e raccolta predefiniti

Esporta i dati utilizzando cbexport json per creare un dump JSON di tutte le raccolte nel bucket. Per l'`--format`opzione puoi usare o. `lines` `list`

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

Importa i dati in una raccolta Amazon DocumentDB utilizzando mongoimport con l'opzione appropriata per importare le righe o l'elenco:

linee:

```
mongoimport \
  --db <database> \
  --collection <collection> \
```

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

elenco:

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

Da raccolte Couchbase a raccolte Amazon DocumentDB

Esporta i dati utilizzando cbexport json per creare un dump JSON di ogni raccolta. Usa l'`--include-data`opzione per esportare ogni raccolta. Per l'`--format`opzione puoi usare `lines` `olist`. Usa le `--collection-field` opzioni `--scope-field` e per memorizzare il nome dell'ambito e della raccolta nei campi specificati in ogni documento JSON.

```
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"
```

Poiché cbexport ha aggiunto i `_collection` campi `_scope` and a ogni documento esportato, puoi rimuoverli da ogni documento nel file di esportazione tramite cerca e sostituisci o qualsiasi metodo tu preferisca. `sed`

Importa i dati di ogni raccolta in una raccolta Amazon DocumentDB utilizzando mongoimport con l'opzione appropriata per importare le righe o l'elenco:

linee:

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

elenco:

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

Migrazione online

Prendi in considerazione una migrazione online quando hai bisogno di ridurre al minimo i tempi di inattività e le modifiche in corso devono essere replicate su Amazon DocumentDB quasi in tempo reale.

Scopri come eseguire una migrazione in tempo reale da Couchbase ad Amazon DocumentDB per scoprire come eseguire una migrazione in tempo reale ad Amazon DocumentDB. La documentazione ti guida nella distribuzione della soluzione e nell'esecuzione di una migrazione in tempo reale da un bucket a un cluster Amazon DocumentDB.

Argomenti

- Couchbase Server 6.x o versioni precedenti
- Couchbase Server 7.0 o successivo

Couchbase Server 6.x o versioni precedenti

Dal bucket Couchbase alla raccolta Amazon DocumentDB

L'utilità di migrazione per Couchbase è preconfigurata per eseguire una migrazione online di un bucket Couchbase verso una raccolta Amazon DocumentDB. Osservando la configurazione del connettore sink, il `document.id.strategy` parametro è configurato per utilizzare il valore della chiave del messaggio come valore del `_id` campo (vedi Sink Connector Id Strategy Properties):

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

Couchbase Server 7.0 o successivo

Bucket Couchbase con ambito e raccolta predefiniti

L'utilità di migrazione per Couchbase è preconfigurata per eseguire una migrazione online di un bucket Couchbase verso una raccolta Amazon DocumentDB. Osservando la configurazione del connettore sink, il `document.id.strategy` parametro è configurato per utilizzare il valore della chiave del messaggio come valore del `_id` campo (vedi Sink Connector Id Strategy Properties):

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

Da raccolte Couchbase a raccolte Amazon DocumentDB

Configura il connettore di origine per trasmettere ogni raccolta Couchbase in ciascun ambito a un argomento separato (vedi Opzioni di configurazione del codice sorgente). Ad esempio:

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

Configura il connettore sink per lo streaming da ogni argomento a una raccolta Amazon DocumentDB separata (vedi Sink Connector Configuration Properties). Ad esempio:

```
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>'
```

Convalida

Questa sezione fornisce un processo di convalida dettagliato per verificare la coerenza e l'integrità dei dati dopo la migrazione ad Amazon DocumentDB. Le fasi di convalida si applicano indipendentemente dal metodo di migrazione.

Argomenti

- Verifica che tutte le raccolte esistano nella destinazione
- Verifica il numero di documenti tra i cluster di origine e di destinazione
- Confronta i documenti tra cluster di origine e di destinazione

Verifica che tutte le raccolte esistano nella destinazione

Fonte Couchbase

opzione 1: query workbench

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

opzione 2: strumento cbq

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

obiettivo Amazon DocumentDB

mongosh (vedi Connessione al tuo cluster Amazon DocumentDB):

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

Verifica il numero di documenti tra i cluster di origine e di destinazione

Fonte Couchbase

Couchbase Server 6.x o versioni precedenti

opzione 1: query workbench

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

opzione 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 o successivo

opzione 1: query workbench

```
SELECT COUNT(*)
```

```
FROM `<bucket>`.`<scope>`.`<collection>`
```

opzione 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>`"
```

Obiettivo Amazon DocumentDB

mongosh (vedi Connessione al tuo cluster Amazon DocumentDB):

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

Confronta i documenti tra cluster di origine e di destinazione

Fonte Couchbase

Couchbase Server 6.x o versioni precedenti

opzione 1: query workbench

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

opzione 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 o successivo

opzione 1: query workbench

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

opzione 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"
```

Obiettivo Amazon DocumentDB

mongosh (vedi Connessione al tuo cluster Amazon DocumentDB):

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

Date di supporto delle versioni del motore Amazon DocumentDB

È possibile utilizzare le date seguenti per pianificare i cicli di test e aggiornamento.

Calendario di rilascio per le versioni principali di Amazon DocumentDB

Amazon DocumentDB attualmente supporta le seguenti versioni principali. Per aggiornare la versione del motore, consulta Aggiornamento di una versione principale del motore oAggiornamento tramite AWS DMS.

Versione del motore	Data di rilascio	Data di fine del supporto standard	Inizio dell'Extended Support (prezzo del primo anno)	Inizio dell'Extended Support (prezzo per il terzo anno)	Fine dell'Extended Support
Versione 3.6	9 gennaio 2019	30 marzo 2026	31 marzo 2026	31 marzo 2028	30 marzo 2029
Versione 4.0	9 novembre 2020	N/A	N/A	N/A	N/A
Versione 5.0	1 marzo 2023	N/A	N/A	N/A	N/A
Versione 8.0	14 novembre 2025	N/A	N/A	N/A	N/A

Amazon DocumentDB supporta più versioni del motore nell'ambito del supporto standard. Puoi continuare a utilizzare una versione oltre la data di fine del supporto standard pagando una tariffa di Extended Support. Per ulteriori informazioni, consulta la pagina dei Supporto esteso Amazon DocumentDB prezzi di Amazon DocumentDB.

Calendario di rilascio per le versioni secondarie di Amazon DocumentDB

Amazon DocumentDB attualmente supporta le seguenti versioni secondarie. Il programma di rilascio varierà a seconda delle funzionalità o delle correzioni aggiuntive. Le versioni secondarie possono raggiungere la fine del supporto standard prima che lo facciano le versioni principali corrispondenti. Per aggiornare la versione del motore, consultaAggiornamento di una versione secondaria del motore.

> **Note**
>
> Le versioni secondarie sono disponibili a partire dalla versione principale 5.0.

Versione del motore	Data di rilascio
Versione 5.0.0 (LTS)	16 febbraio 2026
Versione 5.0.1	8 giugno 2026

Amazon DocumentDB designa alcune versioni come versioni Long-Term Support (LTS). Per ulteriori informazioni, consulta Uso di versioni con supporto a lungo termine (Long-Term Support, LTS).

Aggiornamento della versione principale in-place di Amazon DocumentDB

Puoi eseguire un aggiornamento della versione principale (MVU) sul posto del tuo cluster Amazon DocumentDB mantenendo gli stessi endpoint, storage e tag. Le tue applicazioni continuano a funzionare senza modifiche. Questa funzionalità è disponibile senza costi aggiuntivi Regioni AWS ovunque sia disponibile Amazon DocumentDB.

> **Important**
>
> Il cluster non è disponibile durante l'aggiornamento e si riavvia più volte. Non connettere, leggere o scrivere sul cluster dopo aver avviato l'aggiornamento. Il tempo di inattività varia in base al numero di raccolte, indici, database e istanze. Eseguite l'aggiornamento durante la finestra di manutenzione o durante le ore di scarso utilizzo.
>
> Una volta eseguito l'aggiornamento, non è possibile effettuare il downgrade a una versione precedente. Se necessario, puoi ripristinare lo snapshot precedente all'aggiornamento su un nuovo cluster.

Argomenti

- Percorsi di aggiornamento supportati
- Prerequisiti
- Best practice
- Cosa cambia dopo l'aggiornamento ad Amazon DocumentDB 8.0
- Post-upgrade considerazioni per i cluster aggiornati dalla versione 3.6 o 4.0
- Esecuzione dell'aggiornamento

- Risoluzione dei problemi

Percorsi di aggiornamento supportati

La tabella seguente elenca le versioni del motore di destinazione che è possibile selezionare per ciascuna versione di origine. Non è necessario eseguire prima l'aggiornamento alla versione `.0` secondaria e quindi applicare un aggiornamento della versione secondaria.

Versione di origine	Versioni supportate per la destinazione
3.6	Qualsiasi versione secondaria 5.0 o 8.0 pubblicata
4.0	Qualsiasi versione secondaria 5.0 o 8.0 pubblicata
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

Per considerazioni successive all'aggiornamento, vedere e. the section called “Post-upgrade considerazioni per i cluster aggiornati dalla versione 3.6 o 4.0” the section called “Cosa cambia dopo l'aggiornamento ad Amazon DocumentDB 8.0”

Per visualizzare le versioni secondarie disponibili nella tua AWS regione, usa il AWS CLI comando. `aws docdb describe-db-engine-versions` Per un elenco delle versioni secondarie rilasciate, consultaNote di rilascio.

> **Note**
>
> In-place MVU non è supportato per i cluster globali o i cluster elastici. Per aggiornare un cluster globale, rimuovi i cluster secondari, converti il cluster primario in un cluster regionale, esegui l'MVU, quindi ricrea il cluster globale aggiungendo cluster secondari con gli stessi nomi per conservare gli endpoint. Dovrai sostenere dei I/O costi mentre il primario aggiornato replica i dati sui nuovi secondari. Per informazioni dettagliate sulle fasi, consulta Rimuovere un cluster da un cluster globale Amazon DocumentDB.

Prerequisiti

> **Important**
>
> Aumenta la scalabilità delle istanze burstable prima dell'aggiornamento. Se il tuo cluster utilizza tipi di istanza burstable (ad esempio, `db.t3.medium` or`db.t4g.medium`), aumenta l'istanza primaria almeno `db.r5.large` o `db.r6g.large` prima di iniziare l'aggiornamento. Le istanze burstable potrebbero non disporre di CPU e memoria sufficienti per completare il processo di aggiornamento, con conseguenti errori di aggiornamento e prolungata indisponibilità del cluster. È possibile ridimensionare il sistema una volta completato l'aggiornamento.
>
> Aumenta la scalabilità dell'istanza serverless writer prima dell'aggiornamento.
> Se il cluster ha un'istanza di writer serverless, la DCU massima in `ServerlessV2ScalingConfiguration` deve essere impostata su 3 o superiore prima di iniziare l'aggiornamento. I cluster con una DCU massima inferiore a 3 non possono essere aggiornati perché non dispongono di CPU o memoria sufficienti per completare l'aggiornamento.
>
> Controlla gli indici parziali prima dell'aggiornamento (aggiornamento da Amazon DocumentDB 5.0 a 8.0). Prima di iniziare l'aggiornamento, controlla i tuoi indici parziali per verificare `partialFilterExpression` che l'`$type`operatore corrisponda a determinati tipi di BSON: in particolare gli alias `binData`a`bool`,`date`, `double`e`decimal`, e `array` di tipo o un codice di tipo numerico come. `{ "$type" : 1 }` Tali indici parziali causano il fallimento dell'aggiornamento, anche se l'indice funziona normalmente per le letture e le scritture. `db.collection.getIndexes()`Usalo per rivedere i tuoi `partialFilterExpression` indici parziali. Elimina l'indice parziale interessato prima di iniziare l'aggiornamento e ricrealo al termine dell'aggiornamento. Se non sei sicuro che un indice sia interessato, contatta l' AWS assistenza prima dell'aggiornamento.

- Tipo di istanza: Amazon DocumentDB 4.0+ non supporta le istanze db.r4. Modifica qualsiasi `db.r4.*` istanza in istanze o più recenti prima dell'aggiornamento. `db.r5.*` Consulta Modifica di un'istanza Amazon DocumentDB e Classi di istanze supportate per regione.
- Patch del sistema operativo: applica eventuali azioni di manutenzione del sistema operativo in sospeso su tutte le istanze prima dell'aggiornamento. Consulta Aggiornamenti del sistema operativo Amazon DocumentDB.

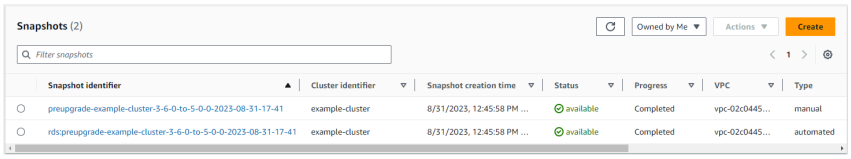
> **Note**
>
> Le patch del motore a livello di cluster in sospeso possono nascondere le patch del sistema operativo dell'istanza. Se necessario, applica prima le patch del motore. Consulta Esecuzione di un aggiornamento della patch alla versione del motore di un cluster.

- Limiti dell'indice sulle istanze burstable (istanze della famiglia t): se hai più di 3.000 indici sulle istanze burstable, aumenta il primario ad almeno db.r5.large prima dell'aggiornamento. Puoi ridimensionare la scala al termine dell'aggiornamento.

Istanza	Indici massimi per MVU
db.t4g.medium	3K
db.t3.medium	10.000

- Gruppo di parametri: prepara un gruppo di parametri del cluster personalizzato per la versione di destinazione prima dell'aggiornamento. Se non ne viene specificato uno, verrà utilizzato il gruppo di parametri predefinito per la versione di destinazione (ad esempio, `default.docdb5.0` o`default.docdb8.0`).
- Istantanea manuale: crea un'istantanea manuale prima dell'aggiornamento. Il processo di aggiornamento crea un'istantanea automatica denominata`preupgrade-<name>-<version>-<timestamp>`, ma crea sempre il tuo backup. Consulta Creazione manuale di un'istantanea del cluster.

> **Note**
>
> L'istantanea automatica creata dal processo di aggiornamento non verrà eliminata automaticamente dopo il completamento dell'aggiornamento della versione principale sul posto. Questa istantanea non comporterà alcun costo fintanto che rientra nel periodo di conservazione. Puoi scegliere di eliminare questa istantanea dopo aver verificato l'avvenuto aggiornamento del cluster.



Best practice

Prima dell'aggiornamento: test con un clone

1. Utilizzalo Clonazione di un volume per un cluster Amazon DocumentDB per creare un clone del tuo cluster. Non dovrai sostenere costi di archiviazione a meno che non modifichi i dati sul clone.
2. Abbina il numero di istanze del clone al cluster di destinazione per una stima realistica dei tempi.
3. Esegui l'MVU sul clone e verifica completamente le differenze funzionali.
4. Controlla se è già programmato un aggiornamento eseguendo `describe-db-clusters` e cercando. `PendingModifiedValues.EngineVersion` Se hai modificato il cluster e hai scelto di applicarlo nella successiva finestra di manutenzione, la pianificazione non sarà visibile nella console ma potrai visualizzarla in AWS CLI:

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

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

5. Se il test ha esito positivo, procedi con l'aggiornamento del cluster di produzione.

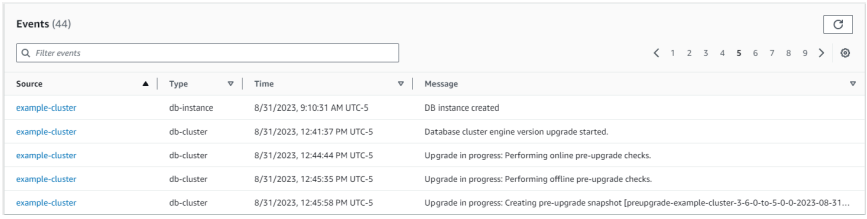
Durante l'aggiornamento

È possibile monitorare lo stato di avanzamento dell'aggiornamento della versione principale in corso sottoscrivendo gli eventi di manutenzione del cluster. Al termine dell'aggiornamento, riceverai l'evento «La versione principale del cluster di database è stata aggiornata». Questo e altri eventi che si verificano durante l'aggiornamento vengono visualizzati nella sezione Eventi e tag della pagina dei dettagli del cluster nella console Amazon DocumentDB. Lo stato del cluster cambia quindi da `upgrading`. `available`

Durante l'aggiornamento vengono generati i seguenti eventi:

1. È iniziato l'aggiornamento della versione principale del motore del cluster di database. Il cluster rimane online.
2. Preparazione dell'aggiornamento in corso: avvio dei controlli preliminari di aggiornamento online.
3. Preparazione dell'aggiornamento in corso: completamento dei controlli preliminari di aggiornamento online.
4. Mettere offline il cluster di database mentre l'istanza principale completa il patch/upgrade processo.
5. Preparazione dell'aggiornamento in corso: avvio dei controlli preliminari dell'aggiornamento offline.
6. Preparazione dell'aggiornamento in corso: controlli preliminari per l'aggiornamento offline completati.
7. <version-to><timestamp>Aggiornamento in corso: creazione di un'istantanea precedente all'aggiornamento [preupgrade- <cluster-name>- <version-from>-to- -].
8. Aggiornamento in corso: clonazione del volume.
9. Aggiornamento in corso: aggiornamento di writer.
10. Aggiornamento in corso: aggiornamento dei lettori.
11. La versione principale del motore del cluster database è stata aggiornata.

Gli eventi sono visibili nella console nella pagina Eventi:



Da AWS CLI, è possibile eseguire `aws docdb describe-events` per monitorare l'avanzamento dell'aggiornamento. Per ricevere notifiche automaticamente, utilizza l'`aws docdb create-event-subscription`opzione per iscriverti agli eventi e indirizzarli ad Amazon SNS per e-mail, messaggi push o altri metodi di consegna. Per ulteriori informazioni, consulta Iscrizione agli eventi di Amazon DocumentDB.

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

Il comando restituisce un output simile al seguente:

```
{
    "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"
        }
    ]
}
```

Dopo l'aggiornamento

> **Warning**
>
> Subito dopo l'aggiornamento della versione principale sul posto, il cluster Amazon DocumentDB ripopola i metadati degli indici utilizzati dal motore del database per ottimizzare i piani di esecuzione delle query. Le prestazioni delle query tornano ai livelli previsti una volta completato questo processo. In genere termina in pochi minuti, ma può richiedere fino a due ore a seconda del numero di indici sul cluster.
> Non riavviate, eseguite il failover o scalate l'istanza di writer durante questo periodo, poiché ciò potrebbe interrompere up/down il ricalcolo dei metadati dell'indice. Attendi di osservare le prestazioni previste delle query prima di apportare tali modifiche.
> Tieni traccia dei progressi tramite i seguenti eventi del cluster:
>
> - Post-upgrade stato del cluster: è iniziato il processo di aggiornamento dei metadati dell'indice
> - Post-upgrade stato del cluster: processo di aggiornamento dei metadati dell'indice completato in X secondi

Contatta l' AWS assistenza se il processo di aggiornamento dei metadati dell'indice non viene completato entro tre ore o se continui a riscontrare problemi di prestazioni dopo il completamento del processo.

1. Scatta un'istantanea manuale del cluster aggiornato nel caso in cui sia necessario ripristinare lo stato successivo all'aggiornamento. Il processo di istantanea automatica riprenderà non appena sarà completato l'aggiornamento della versione principale sul posto.
2. Cluster di tag aggiornati dalla versione 3.6. Aggiungi un tag per differenziare i cluster aggiornati dalla versione 3.6. (vedi). the section called “Post-upgrade considerazioni per i cluster aggiornati dalla versione 3.6 o 4.0”
3. Aggiorna il driver. Per utilizzare nuove funzionalità (ad esempio, confronto, visualizzazioni o compressione Zstd in 8.0), esegui l'aggiornamento alla versione corrispondente dell'API MongoDB. Per ulteriori informazioni, consulta Cosa c'è di nuovo in Amazon DocumentDB 8.0.
4. Esegui un test approfondito. Convalida la tua applicazione rispetto al cluster aggiornato.

Cosa cambia dopo l'aggiornamento ad Amazon DocumentDB 8.0

Dopo aver eseguito un aggiornamento della versione principale ad Amazon DocumentDB 8.0 (da Amazon DocumentDB 5.0 o direttamente da 3.6 o 4.0), le seguenti funzionalità sono abilitate o modificate:

- Fascicolazione. Amazon DocumentDB 8.0 supporta la raccolta. Dopo l'aggiornamento, le regole di confronto sono abilitate per impostazione predefinita per le nuove raccolte e i relativi indici e per i nuovi indici sulle raccolte esistenti.
- Indice di testo. I nuovi indici di testo vengono creati utilizzando Text Index V2, che utilizza un parser di ricerca di testo aggiornato per una migliore compatibilità con MongoDB. Gli indici di testo esistenti non sono interessati.
- Versione Query Planner. Se non disponi di un gruppo di parametri personalizzato, viene creato un nuovo gruppo di parametri predefinito per Amazon DocumentDB 8.0 con Planner versione 3 selezionata automaticamente. Con la versione 3 di Query Planner, sono disponibili anche le visualizzazioni.
- Compressione. Amazon DocumentDB 8.0 supporta la compressione dei documenti basata su dizionari utilizzando l'algoritmo Zstd. Dopo l'aggiornamento, vengono create nuove raccolte con

la compressione Zstd abilitata per impostazione predefinita. Le raccolte esistenti a partire dalla versione 5.0 mantengono le impostazioni di compressione. Per sfruttare la compressione Zstd sulle raccolte esistenti, puoi modificarne le impostazioni di compressione. Per ulteriori informazioni, consulta Gestione della compressione basata su dizionari in Amazon DocumentDB 8.0.

- Ricostruzione dell'indice. Se si esegue l'aggiornamento da Amazon DocumentDB 5.0 ad Amazon DocumentDB 8.0, non è necessaria la ricostruzione dell'indice. Se stai effettuando l'aggiornamento direttamente da Amazon DocumentDB 3.6 o 4.0 a 8.0, ricostruisci gli indici come descritto in. the section called “Post-upgrade considerazioni per i cluster aggiornati dalla versione 3.6 o 4.0”

> **Important**
>
> Amazon DocumentDB 8.0 richiede TLS 1.2 o versioni successive. TLS 1.0 e TLS 1.1 non sono più supportati.

> **Note**
>
> Per un elenco completo delle differenze funzionali, consulta. Compatibilità di Amazon DocumentDB con MongoDB

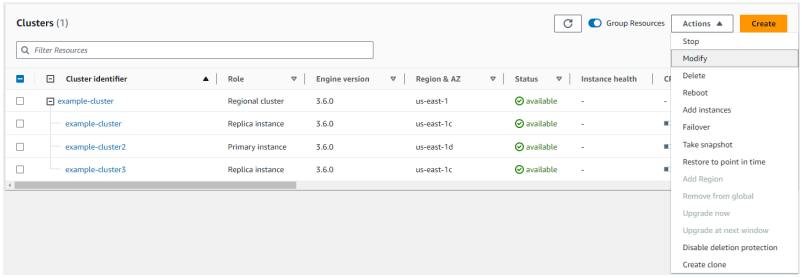
Post-upgrade considerazioni per i cluster aggiornati dalla versione 3.6 o 4.0

- Ricostruzione dell'indice. Un MVU mantiene gli indici originali. Amazon DocumentDB 5.0 e versioni successive hanno migliorato la manutenzione degli indici e la garbage collection, in particolare per gli indici a bassa cardinalità. Dopo l'aggiornamento da 3.6 o 4.0 (a 5.0 o 8.0), ricostruisci gli indici per garantire prestazioni ottimali delle query (opzionale, comporta l'aggiunta di ulteriori). I/O Consulta Manutenzione dell'indice utilizzando ReIndex .
- Confronto numerico tra documenti secondari (solo 3.6). I cluster aggiornati dalla versione 3.6 ereditano il comportamento 3.6 in cui i tipi numerici nei documenti secondari non vengono confrontati tra i tipi. Ad esempio, `{a: {b: NumberLong(1)}}` non è uguale `{a: {b: 1}}` in 3.6, ma è uguale in 4.0 e versioni successive. Questo comportamento influisce su tutti i cluster aggiornati dalla versione 3.6.

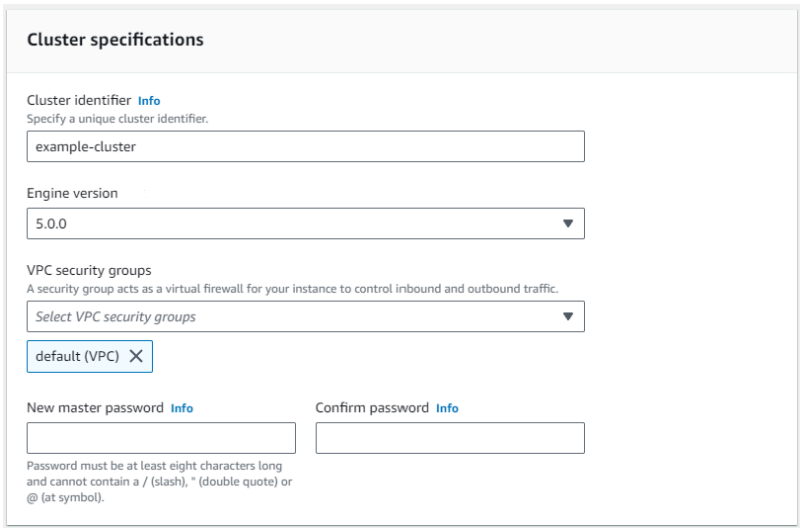
Esecuzione dell'aggiornamento

Using the Console di gestione AWS

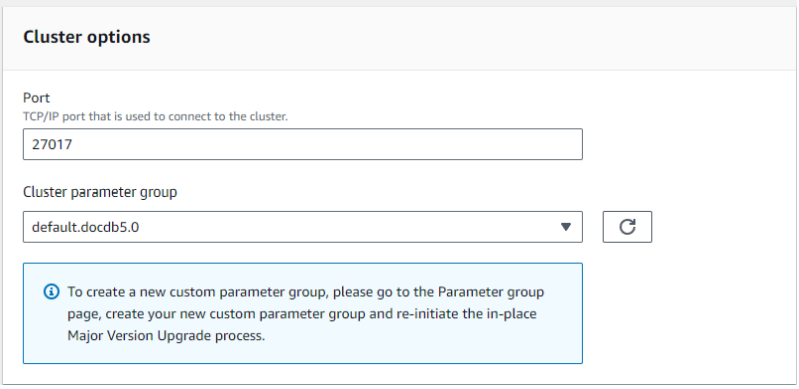
1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.
2. Nella tabella Clusters, seleziona il cluster di origine, scegli Azioni, quindi Modifica.



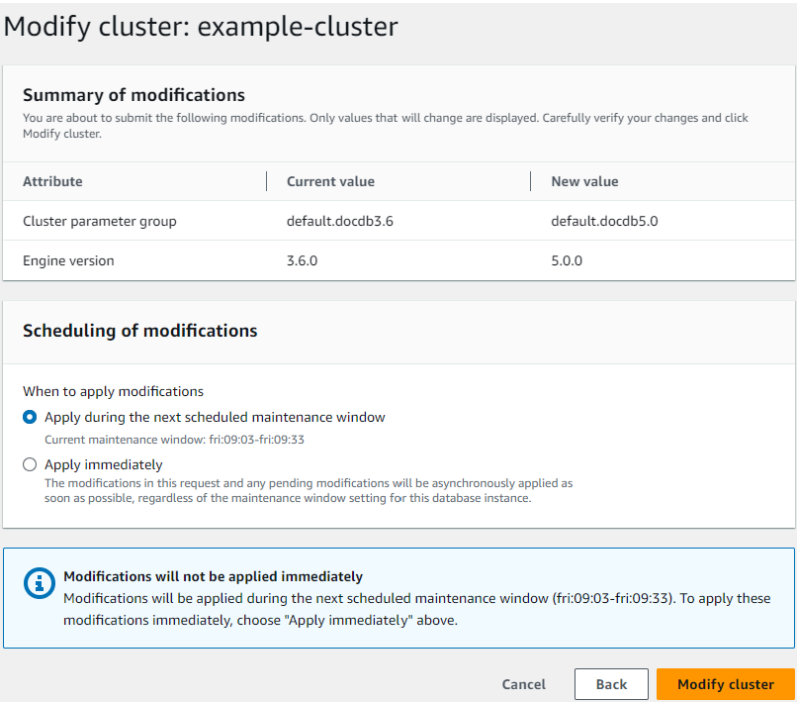
3. Nelle specifiche del cluster, scegli la versione di destinazione (ad esempio, 5.0.0 o 8.0.0) dal menu a discesa della versione del motore.



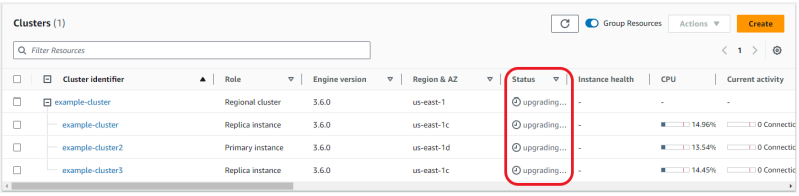
4. Nelle opzioni del cluster, seleziona il gruppo di parametri del cluster per la versione del motore di destinazione. Puoi utilizzare il gruppo di parametri predefinito (ad esempio, default.docdb5.0 o default.docdb8.0) o un gruppo di parametri personalizzato che hai creato.



5. Scegli Continua, seleziona la tua preferenza di pianificazione (applica immediatamente o nella prossima finestra di manutenzione), quindi scegli Modifica cluster.



6. Monitora lo stato del cluster nella tabella dei cluster mentre passa all'aggiornamento:



Using the AWS CLI

Usa [modify-db-cluster](#) con la bandiera: `--allow-major-version-upgrade`

```
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
```

Sostituisci ciascuno *placeholder* con le informazioni del tuo cluster.

Risoluzione dei problemi

- Pre-upgrade errore di controllo. Prima dell'inizio dell'aggiornamento, Amazon DocumentDB esegue controlli di convalida pre-aggiornamento. Le seguenti sono le cause più comuni di errori di pre-controllo:
  - L'aggiornamento non può procedere perché le raccolte hanno nomi con 58 o più caratteri. Rinomina le raccolte interessate con nomi più brevi prima di riprovare l'aggiornamento.
  - L'aggiornamento non può procedere perché il numero di indici supera il limite per il tipo di istanza. Esegui l'upgrade a un tipo di istanza più grande prima di riprovare l'aggiornamento. Per i limiti dell'indice per tipo di istanza, consulta. Prerequisiti
- Errore di aggiornamento e rollback. Se l'aggiornamento non riesce, tenta automaticamente un rollback. Un rollback riuscito genera l'evento: «Il cluster di database è in uno stato che non può essere aggiornato». Il cluster torna allo stato precedente all'aggiornamento e puoi continuare a utilizzarlo. Contatta l' AWS assistenza per risolvere il problema prima di riprovare.
- Post-upgrade prestazioni. Durante l'esecuzione dell'aggiornamento dei metadati dell'indice possono verificarsi un temporaneo peggioramento delle prestazioni e un elevato utilizzo della CPU. Se il deterioramento persiste per oltre 3 ore, contatta l'assistenza. AWS

Per ulteriore assistenza, contatta. Supporto AWS

Aggiornamento della versione secondaria di Amazon DocumentDB

> **Note**
>
> Questo argomento tratta gli aggiornamenti delle versioni minori all'interno della stessa versione principale (ad esempio, l'aggiornamento da 5.0.0 a 5.0.1). Per informazioni

sull'aggiornamento a una versione principale diversa, consulta. Aggiornamento della versione principale in-place di Amazon DocumentDB

Puoi aggiornare la versione secondaria del motore del tuo cluster Amazon DocumentDB (ad esempio, da 5.0.0 a 5.0.1) modificando il cluster. Gli aggiornamenti di versione minori possono essere applicati immediatamente o durante la finestra di manutenzione successiva.

Operazioni preliminari all'aggiornamento a una versione secondaria

Prima di eseguire un aggiornamento di una versione secondaria, possono essere utili le seguenti azioni:

- Esegui l'aggiornamento durante un periodo di traffico ridotto per ridurre l'impatto di eventuali brevi interruzioni.
- Verifica che la versione secondaria di destinazione sia una destinazione di aggiornamento valida per la versione corrente utilizzando il `describe-db-engine-versions` comando.
- Per i cluster globali, i cluster secondari devono essere aggiornati prima del cluster primario. Per ulteriori informazioni, consulta Applicazione di patch ai cluster globali.

Important

- Gli upgrade sono operazioni unidirezionali: non è possibile effettuare il downgrade dopo l'aggiornamento.
- Il cluster subirà un breve periodo di inattività durante il processo di aggiornamento.

Esecuzione di un aggiornamento a una versione secondaria

Using the Console di gestione AWS

Per eseguire un aggiornamento di una versione minore utilizzando: Console di gestione AWS

1. Accedi Console di gestione AWSe apri la console Amazon DocumentDB.
2. Nella tabella Cluster, seleziona il cluster che desideri aggiornare, scegli Azioni, quindi scegli Modifica.

3. Nella finestra di dialogo Modifica cluster, nella sezione Specifiche del cluster, scegli la versione secondaria di destinazione dal menu a discesa Versione del motore.
4. Scorri verso il basso e scegli Continua.
5. Nella sezione Pianificazione delle modifiche, scegli di applicare la modifica immediatamente o durante la finestra di manutenzione successiva.

   Scegli Modify cluster (Modifica cluster).

Using the AWS CLI

Utilizzate il `modify-db-cluster`comando con il `--engine-version` parametro:

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

Nell'esempio precedente, sostituisci ciascuno di essi *`user input placeholder`* con le informazioni del tuo cluster.

> **Note**
>
> Per applicare l'aggiornamento durante la finestra di manutenzione successiva anziché immediatamente, utilizzate `--no-apply-immediately` al posto di`--apply-immediately`.

Verifica dell'aggiornamento della versione secondaria

Al termine dell'aggiornamento, verificate la versione del motore utilizzando: AWS CLI

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

L'aspetto dell'output di questa operazione è simile al seguente.

```
"5.0.1"
```

Aggiornamento del cluster Amazon DocumentDB tramite AWS Database Migration Service

> **Important**
>
> Amazon DocumentDB non segue gli stessi cicli di vita del supporto di MongoDB e la pianificazione di fine del ciclo di vita di MongoDB non si applica ad Amazon DocumentDB.

Puoi aggiornare il tuo cluster Amazon DocumentDB a una versione superiore con tempi di inattività minimi utilizzando. AWS DMS AWS DMS è un servizio completamente gestito che semplifica la migrazione da versioni precedenti di Amazon DocumentDB, database relazionali e database non relazionali al cluster Amazon DocumentDB di destinazione.

Argomenti

- Fase 1: abilitare i flussi di modifica
- Passaggio 2: modifica la durata di conservazione dei flussi di modifica
- Fase 3: Esegui la migrazione degli indici
- Fase 4: Creare un AWS DMS istanza di replica
- Fase 5: Creare un AWS DMS endpoint di origine
- Fase 6: Creare un AWS DMS endpoint di destinazione
- Fase 7: Creare ed eseguire un'attività di migrazione
- Fase 8: Modifica dell'endpoint dell'applicazione nel cluster Amazon DocumentDB di destinazione

Fase 1: abilitare i flussi di modifica

Per eseguire una migrazione con tempi di inattività minimi, è AWS DMS necessario l'accesso ai flussi di modifica del cluster. I flussi di modifica di Amazon DocumentDB forniscono una sequenza ordinata nel tempo di eventi di aggiornamento che si verificano all'interno delle raccolte e dei database del cluster. La lettura dal flusso di modifiche consente di AWS DMS eseguire l'acquisizione dei dati di modifica (CDC) e applicare aggiornamenti incrementali al cluster Amazon DocumentDB di destinazione.

Per abilitare i flussi di modifica per tutte le raccolte su un database specifico, esegui l'autenticazione nel tuo cluster Amazon DocumentDB utilizzando la shell mongo ed esegui i seguenti comandi:

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

Passaggio 2: modifica la durata di conservazione dei flussi di modifica

Successivamente, modifica il periodo di conservazione del flusso di modifiche in base al periodo di conservazione degli eventi di modifica nel flusso di modifiche. Ad esempio, se prevedi che la migrazione del cluster Amazon DocumentDB richieda 12 ore, devi impostare la conservazione del flusso di modifiche su un valore superiore a 12 ore. AWS DMS Il periodo di conservazione predefinito per il cluster Amazon DocumentDB è di tre ore. Puoi modificare la durata di conservazione dei log del flusso di modifiche per il tuo cluster Amazon DocumentDB in modo che sia compresa tra un'ora e sette giorni utilizzando il Console di gestione AWS . AWS CLI Per ulteriori informazioni, consulta Modifica della durata di conservazione del registro del flusso di modifiche.

Fase 3: Esegui la migrazione degli indici

Crea gli stessi indici sul cluster Amazon DocumentDB di destinazione che hai sul cluster Amazon DocumentDB di origine. Sebbene AWS DMS gestisca la migrazione dei dati, non migra gli indici. Per migrare gli indici, utilizza Amazon DocumentDB Index Tool per esportare gli indici dal cluster Amazon DocumentDB di origine. Puoi ottenere lo strumento creando un clone del repository di GitHub strumenti Amazon DocumentDB e seguendo le istruzioni riportate in. `README.md` Puoi eseguire lo strumento da un'istanza Amazon EC2 o da un AWS Cloud9 ambiente in esecuzione nello stesso Amazon VPC del cluster Amazon DocumentDB.

Nell'esempio seguente, sostituisci ciascuno *`user input placeholder`* con le tue informazioni.

Il codice seguente esegue il dump degli indici dal cluster Amazon DocumentDB di origine:

```
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/
```

Una volta che gli indici sono stati esportati con successo, ripristina tali indici nel cluster Amazon DocumentDB di destinazione. Per ripristinare gli indici esportati nel passaggio precedente, utilizza lo strumento di indicizzazione di Amazon DocumentDB. Il comando seguente ripristina gli indici nel cluster Amazon DocumentDB di destinazione dalla directory specificata.

```
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
```

Per confermare di aver ripristinato correttamente gli indici, connettiti al cluster Amazon DocumentDB di destinazione con la shell mongo ed elenca gli indici per una determinata raccolta. Vedi il codice seguente:

```
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()
```

Fase 4: Creare un AWS DMS istanza di replica

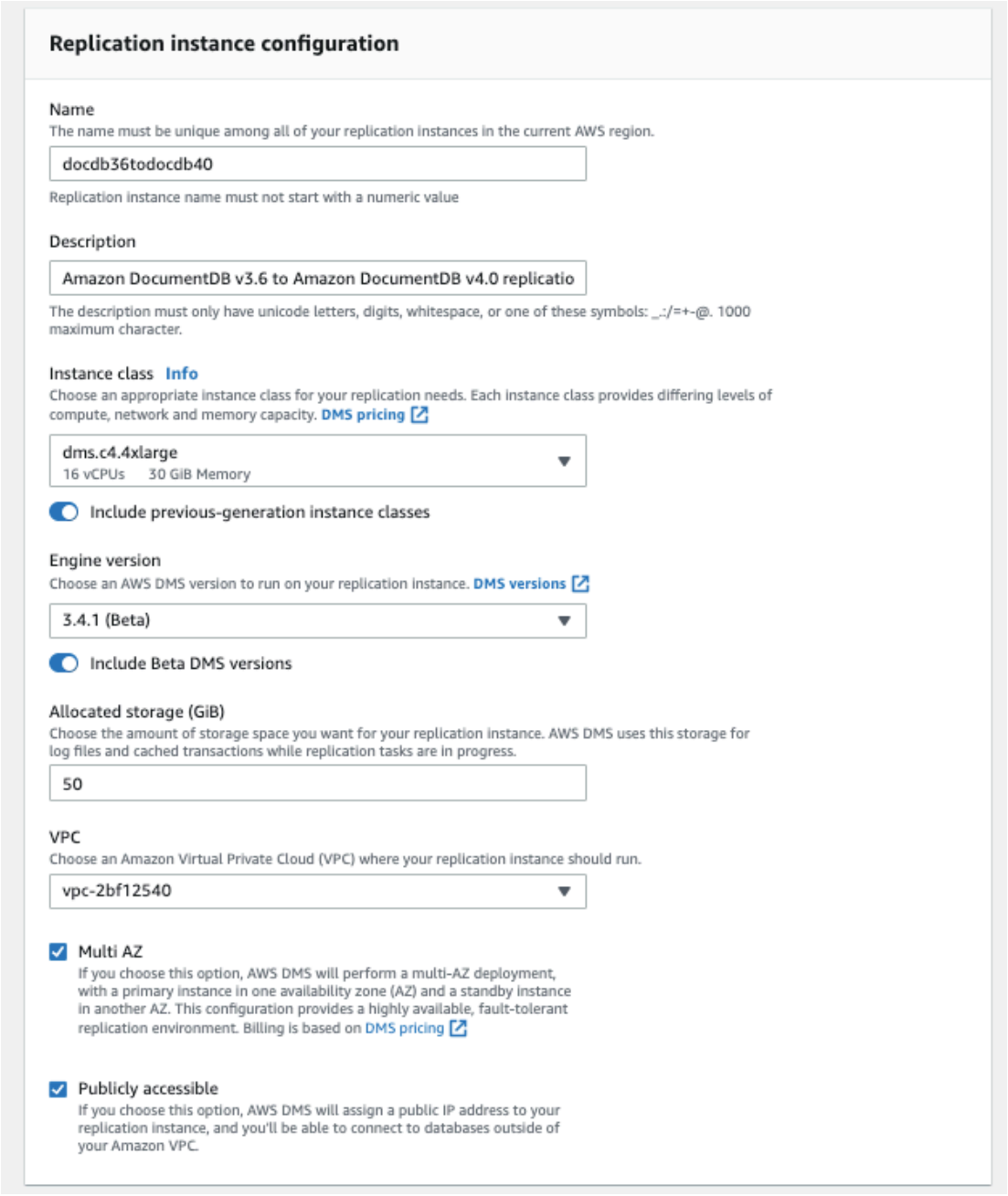
Un'istanza di AWS DMS replica collega e legge i dati dal cluster Amazon DocumentDB di origine e li scrive nel cluster Amazon DocumentDB di destinazione. L'istanza di AWS DMS replica può eseguire sia operazioni di carico di massa che operazioni CDC. La maggior parte di questa elaborazione avviene in memoria. Tuttavia, operazioni di grandi dimensioni potrebbero richiedere un certo buffering su disco. Anche le transazioni e i file di log memorizzati nella cache vengono scritti su disco. Dopo la

migrazione dei dati, l'istanza di replica trasmette anche tutti gli eventi di modifica per assicurarsi che l'origine e la destinazione siano sincronizzate.

Per creare un'istanza di replica AWS DMS :

1. Apri la AWS DMS console.
2. Nel riquadro di navigazione, scegli Replication instances (Istanze di replica).
3. Scegliere Create replication instance (Crea istanza di replica) e immettere le seguenti informazioni:

   - In Nome, inserisci un nome a tua scelta. Ad esempio, `docdb36todocdb40`.
   - In Descrizione, inserisci una descrizione a tua scelta. Per listitem, istanza di replica da Amazon DocumentDB 3.6 ad Amazon DocumentDB 4.0.
   - Per la classe Instance, scegli la dimensione in base alle tue esigenze.
   - Per la versione Engine, scegli `3.4.1.`
   - Per Amazon VPC, scegli Amazon VPC che ospita i cluster Amazon DocumentDB di origine e di destinazione.
   - Per lo storage allocato (GiB), usa il valore predefinito di 50 GiB. Se hai un carico di lavoro con throughput di scrittura elevato, aumenta questo valore in modo che corrisponda al tuo carico di lavoro.
   - Infatti Multi-AZ, scegli Sì se hai bisogno di supporto per l'alta disponibilità e il failover.
   - Per Publicly accessible (Accessibile pubblicamente), abilitare questa opzione.



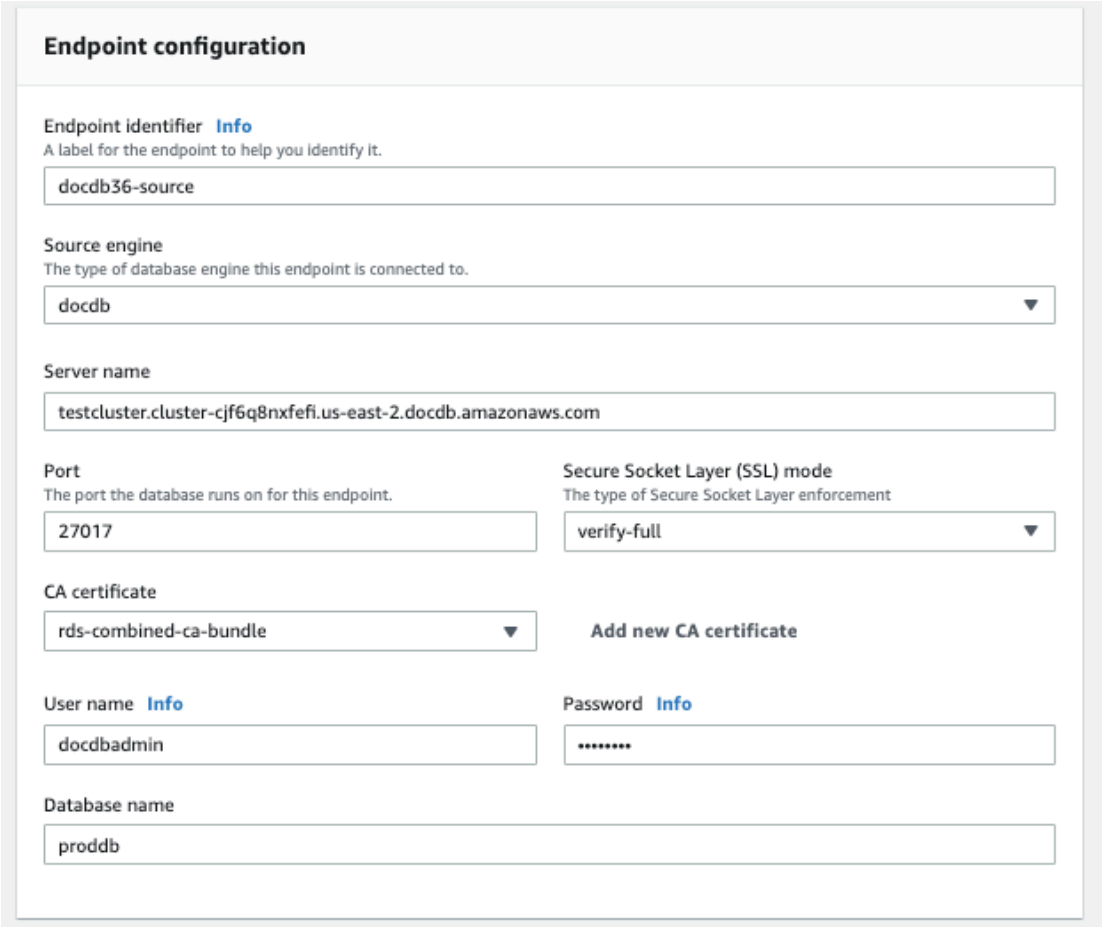
4. Scegli Create replication instance (Crea istanza di replica).

Fase 5: Creare un AWS DMS endpoint di origine

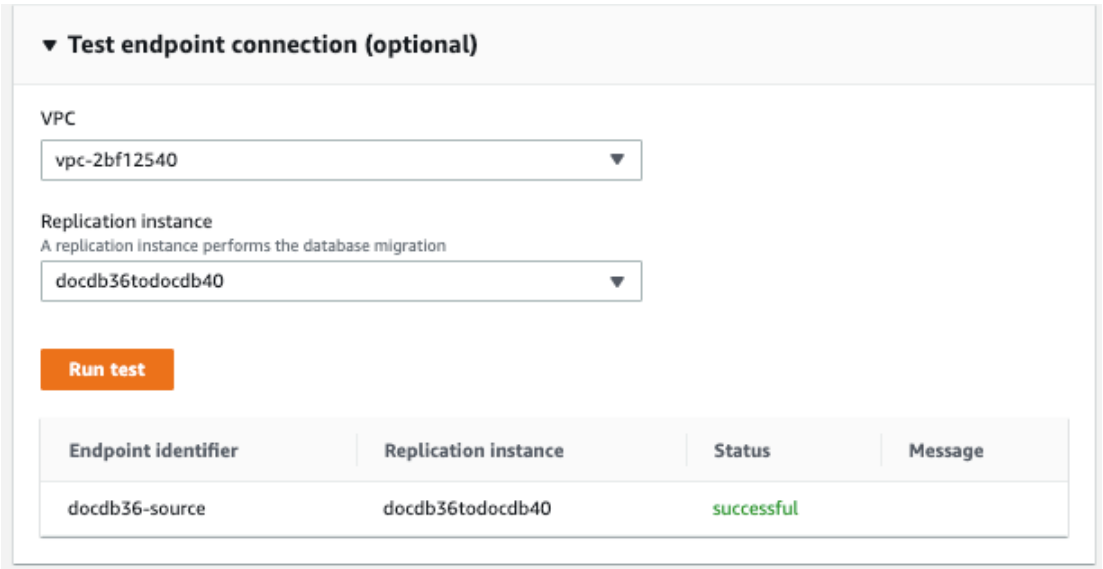
L'endpoint di origine viene utilizzato per il cluster Amazon DocumentDB di origine.

Per creare un endpoint di origine

1. Apri la AWS DMS console.
2. Nel pannello di navigazione, seleziona Endpoint.
3. Scegli `Create endpoint` e inserisci le seguenti informazioni:
   - Per Endpoint type (Tipo di endpoint), scegliere Source (Origine).
   - >Per l'identificatore dell'endpoint, inserisci un nome facile da ricordare, ad esempio. `docdb-source`
   - Per Source engine, scegli. `docdb`
   - Per Nome server, inserisci il nome DNS del cluster Amazon DocumentDB di origine.
   - Per Porta, inserisci il numero di porta del cluster Amazon DocumentDB di origine.
   - Per la modalità SSL, scegli. `verify-full`
   - Per il certificato CA, scegli Aggiungi nuovo certificato CA. Scarica il nuovo certificato CA, nuovo certificato per creare un pacchetto di connessioni TLS. Per l'identificatore del certificato, inserisci. `rds-combined-ca-bundle` Per Import certificate file (Importa file certificato), scegliere Choose file (Seleziona file) e passare al file `.pem` scaricato in precedenza. Selezionare e aprire il file. Scegli Importa certificato, quindi scegli `rds-combined-ca-bundle` dal menu a discesa Scegli un certificato
   - Per Nome utente, inserisci il nome utente principale del cluster Amazon DocumentDB di origine.
   - Per Password, inserisci la password principale del cluster Amazon DocumentDB di origine.
   - Per Nome del database, inserisci il nome del database che desideri aggiornare.



4. Verifica la connessione per verificare che sia stata configurata correttamente.



5. Scegliere Create Endpoint (Crea endpoint).

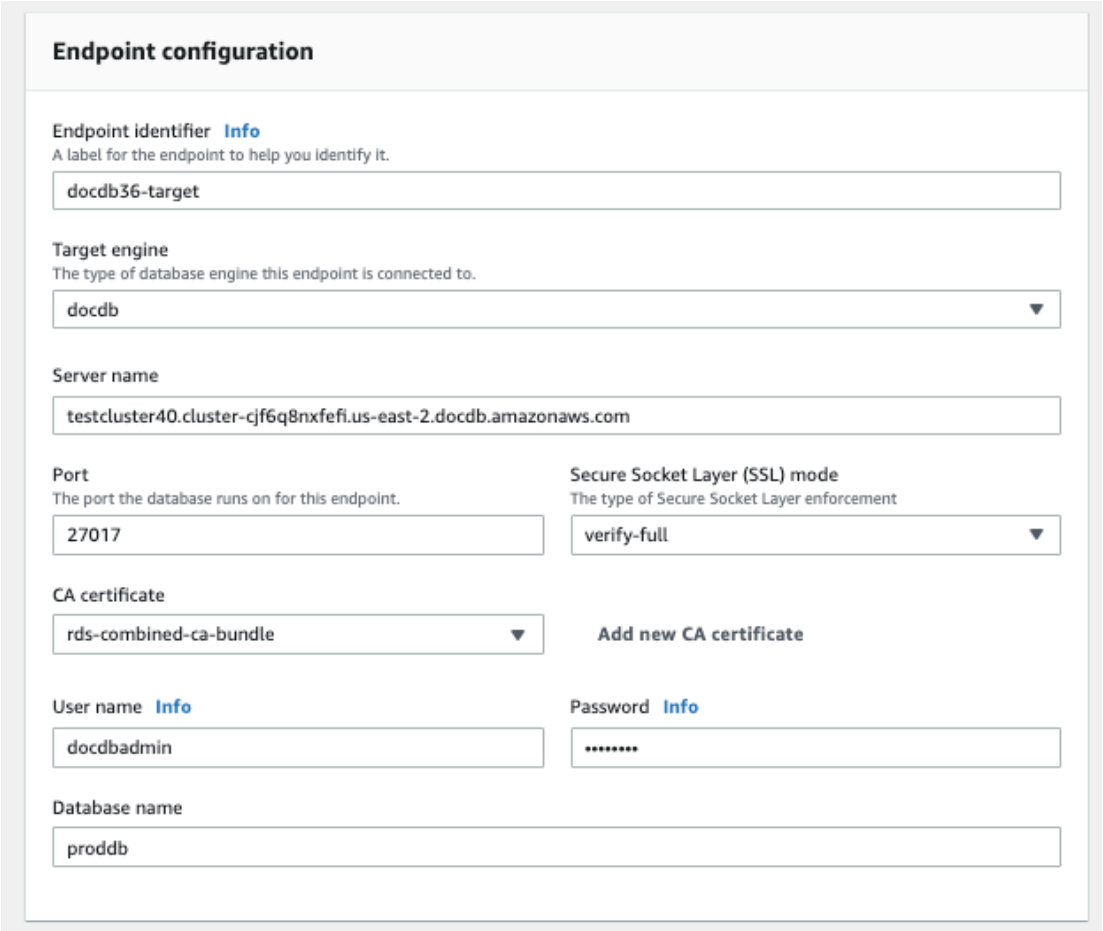
> **Note**
>
> AWS DMS può migrare solo un database alla volta.

Fase 6: Creare un AWS DMS endpoint di destinazione

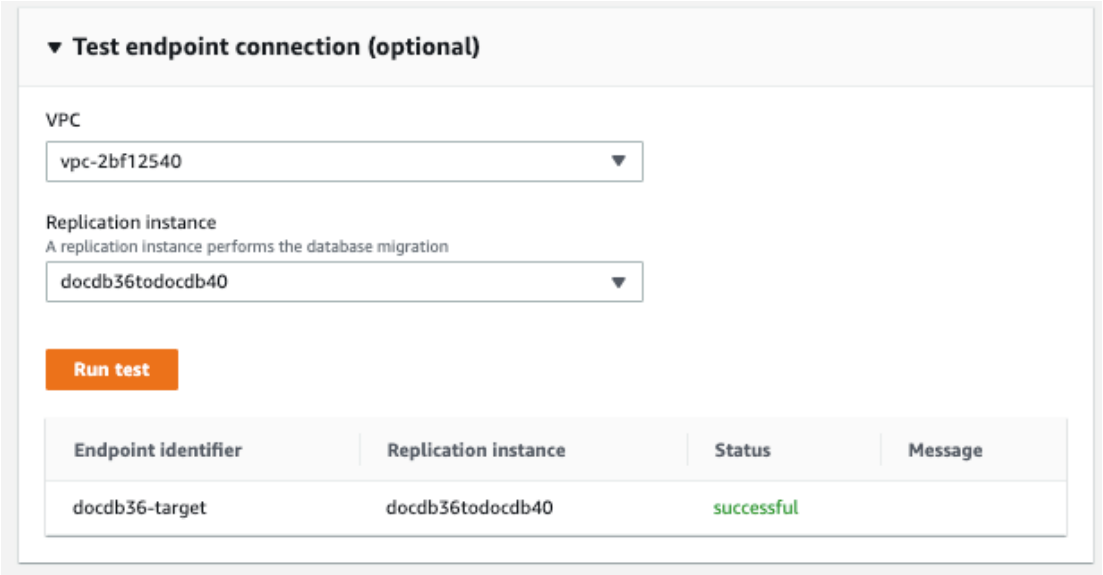
L'endpoint di destinazione è per il cluster Amazon DocumentDB di destinazione.

Per creare un endpoint di destinazione:

1. Apri la AWS DMS console.
2. Nel pannello di navigazione, seleziona Endpoint.
3. Scegliere Create endpoint (Crea endpoint) e immettere le informazioni seguenti:
   - Per Endpoint type (Tipo di endpoint), scegliere Target (Destinazione).
   - Per Endpoint identifier (Identificatore endpoint), immettere un nome facile da ricordare, ad esempio `docdb-target`.
   - Per Source engine, sceglidocdb.
   - Per Nome server, inserisci il nome DNS del cluster Amazon DocumentDB di destinazione.
   - Per Porta, inserisci il numero di porta del cluster Amazon DocumentDB di destinazione.
   - Per la modalità SSL, scegli. `verify-full`
   - Per il certificato CA, scegli il `rds-combined-ca-bundle` certificato esistente dal menu a discesa Scegli un certificato.
   - Per Nome utente, inserisci il nome utente principale del cluster Amazon DocumentDB di destinazione.
   - Per Password, inserisci la password principale del cluster Amazon DocumentDB di destinazione.
   - Per Nome del database, inserisci lo stesso nome di database utilizzato per configurare l'endpoint di origine.



4. Verifica la connessione per verificare che sia stata configurata correttamente.



5. Scegliere Create Endpoint (Crea endpoint).

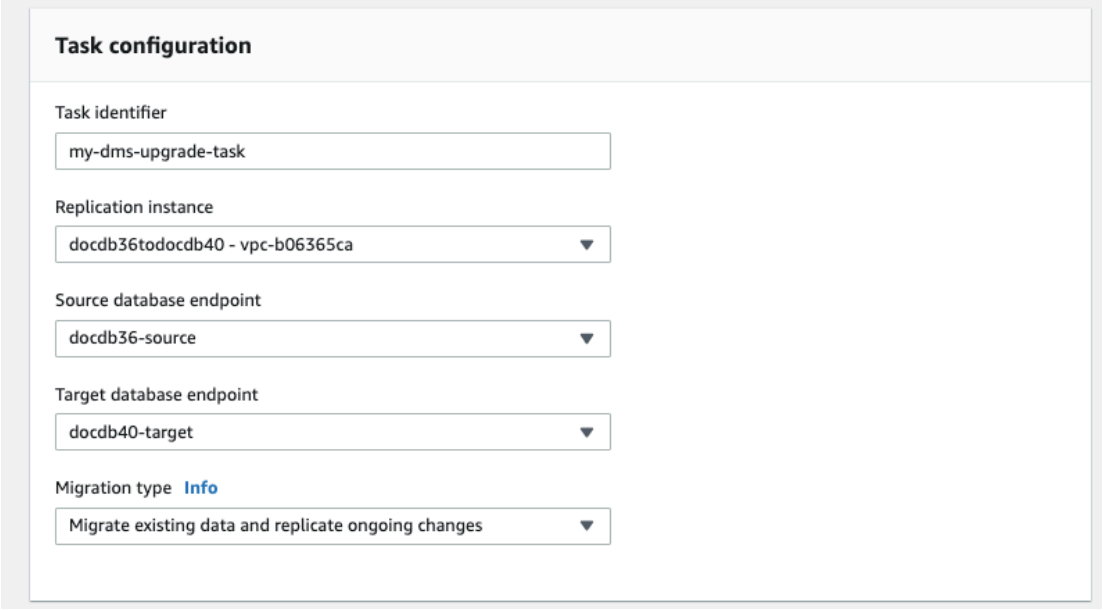
Fase 7: Creare ed eseguire un'attività di migrazione

Un' AWS DMS attività associa l'istanza di replica all'istanza di origine e di destinazione. Quando si crea un'attività di migrazione, si specifica l'endpoint di origine, l'endpoint di destinazione, l'istanza di replica e tutte le impostazioni di migrazione desiderate. È possibile creare un' AWS DMS attività con tre diversi tipi di migrazione: migrazione dei dati esistenti, migrazione dei dati esistenti e replica delle modifiche in corso o replica solo delle modifiche ai dati. Poiché lo scopo di questa procedura dettagliata è aggiornare un cluster Amazon DocumentDB con tempi di inattività minimi, i passaggi utilizzano l'opzione per migrare i dati esistenti e replicare le modifiche in corso. Con questa opzione, AWS DMS acquisisce le modifiche durante la migrazione dei dati esistenti. AWS DMS continua ad acquisire e applicare le modifiche anche dopo il caricamento di grandi quantità di dati. Alla fine i database di origine e di destinazione saranno sincronizzati, con tempi di inattività per la migrazione quasi nulli.

Di seguito sono riportati i passaggi per creare un'attività di migrazione per una migrazione con tempi di inattività minimi:

1. Apri la AWS DMS console.
2. Dal riquadro di navigazione, scegli Attività di migrazione del database.
3. Scegli Crea attività di migrazione del database e inserisci le seguenti informazioni nella sezione Configurazione dell'attività:

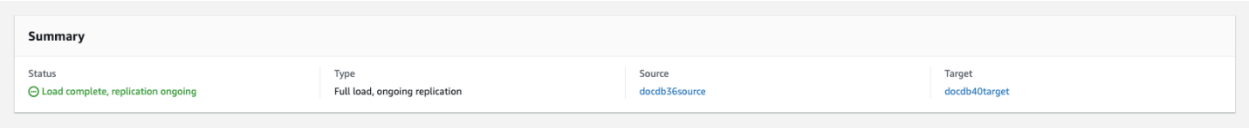
   - Per Task identifier, inserisci un nome facile da ricordare, ad esempio`my-dms-upgrade-task`.
   - Per Descriptive Amazon Resource Name (ARN), inserisci un nome intuitivo per sostituire l'ARN DMS predefinito.
   - Per Replication instance (Istanza di replica), scegliere l'istanza di replica creata in Fase 4: Creare un AWS DMS istanza di replica.
   - Per l'endpoint del database di origine, scegli l'endpoint di origine in cui hai creato. Fase 5: Creare un AWS DMS endpoint di origine
   - Per l'endpoint del database di Target, scegli l'endpoint di destinazione in cui hai creato. Fase 6: Creare un AWS DMS endpoint di destinazione
   - Per il tipo di migrazione, scegli Migra e replica.



4. Inserisci le seguenti informazioni nella sezione Impostazioni dell'attività:

   - Nella sezione Modalità di preparazione della tabella Target, scegli Non fare nulla. Ciò garantirà che gli indici creati nel passaggio 3 non vengano eliminati.
   - Nella sottosezione Registri delle attività, seleziona Attiva i registri. CloudWatch
   - Per la configurazione di avvio dell'attività di migrazione, scegli Automaticamente al momento della creazione. Questo avvierà automaticamente l'attività di migrazione una volta creata.
   - Scegli Crea attività di migrazione del database.

AWS DMS ora inizia la migrazione dei dati dal cluster Amazon DocumentDB di origine al cluster Amazon DocumentDB di destinazione. Lo stato dell'attività dovrebbe cambiare da Avvio a In esecuzione. Puoi monitorare l'avanzamento scegliendo Attività nella AWS DMS console. Dopo diverse minutes/hours operazioni (a seconda delle dimensioni della migrazione), lo stato dovrebbe cambiare da Load complete, replica in corso. Ciò significa che AWS DMS ha completato una migrazione a pieno carico del cluster Amazon DocumentDB di origine verso un cluster Amazon DocumentDB di destinazione e ora sta replicando gli eventi di modifica.



Alla fine la fonte e la destinazione saranno sincronizzate. Puoi verificare se sono sincronizzati eseguendo un'`count()`operazione sulle tue raccolte per verificare che tutti gli eventi di modifica siano stati migrati.

Fase 8: Modifica dell'endpoint dell'applicazione nel cluster Amazon DocumentDB di destinazione

Una volta completato il caricamento completo e dopo la replica continua del processo CDC, sei pronto a modificare l'endpoint di connessione al database dell'applicazione dal cluster Amazon DocumentDB di origine al cluster Amazon DocumentDB di destinazione.

Uso di versioni con supporto a lungo termine (Long-Term Support, LTS)

Amazon DocumentDB designa alcune versioni come versioni Long-Term Support (LTS). I cluster di database che utilizzano LTS possono rimanere sulla stessa versione più a lungo e subire meno cicli di aggiornamento rispetto ai cluster che utilizzano versioni non LTS. Le versioni LTS verranno patchate solo in caso di patch critiche di stabilità e sicurezza senza introdurre nuove funzionalità. Potrebbero essere rilasciate patch opzionali aggiuntive, come metriche per il debug di errori critici. Queste verranno applicate cumulativamente con le patch richieste.

Versione LTS attuale

L'attuale versione LTS per Amazon DocumentDB è:

- Versione secondaria 5.0.0

Per determinare la versione in cui si trova il cluster, consulta. the section called “Come verificare se il cluster della versione 5.0.0 del mio motore esistente è su LTS”

Per ulteriori informazioni sulle tempistiche di supporto e sui cicli di rilascio per le versioni LTS, consulta. Date di supporto delle versioni del motore Amazon DocumentDB Amazon DocumentDB annuncerà la prossima versione LTS attraverso canali di comunicazione standard.

> **Note**
>
> Non è stata ancora designata una versione LTS per Amazon DocumentDB 8.0. Se hai bisogno di LTS, usa la versione del motore 5.0.0 con la versione della patch 3.0.17983 o successiva.

Chi dovrebbe usare LTS

Le versioni di Amazon DocumentDB LTS sono progettate per cluster con finestre di aggiornamento limitate. LTS è ideale per ambienti di produzione in cui la disponibilità del database ha un impatto diretto sulle operazioni aziendali.

Prendi in considerazione le versioni LTS se:

- La vostra applicazione ha requisiti di operatività rigorosi che limitano le opportunità di aggiornamento
- Disponi di tutte le funzionalità e i miglioramenti richiesti dal motore di database per le tue applicazioni

Prendi in considerazione invece le versioni standard se:

- L'applicazione è in grado di gestire interruzioni minime
- Desiderate le funzionalità e i miglioramenti più recenti non appena vengono rilasciati

> **Important**
>
> - Gli upgrade sono operazioni unidirezionali: non è possibile effettuare il downgrade dopo l'aggiornamento
> - Il cluster subirà un breve periodo di inattività durante il processo di aggiornamento

Come creare un nuovo cluster LTS

Per creare un cluster LTS, procedi nel seguente modo:

1. Nella console di gestione, crea un nuovo cluster con la versione 5.0.0 del motore

2. Verifica di utilizzare la versione 5.0.0 del motore dopo la creazione del cluster utilizzando: AWS CLI

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

L'aspetto dell'output di questa operazione è simile al seguente.

```
"5.0.0"
```

Come eseguire l'aggiornamento dalla versione 3.6.0 o 4.0.0 del motore a un cluster LTS 5.0.0

Per eseguire l'aggiornamento a un cluster LTS, procedi nel seguente modo:

1. Nella console di gestione, aggiorna la versione principale (MVU) della versione del motore 3.6.0 o 4.0.0 del cluster alla 5.0.0. Per istruzioni sull'esecuzione di un MVU, vedere. Aggiornamento della versione principale in-place di Amazon DocumentDB
2. Verifica di utilizzare la versione 5.0.0 del motore dopo l'aggiornamento utilizzando: AWS CLI

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

Come verificare se il cluster della versione 5.0.0 del mio motore esistente è su LTS

Per verificare se la versione del cluster 5.0.0 del tuo motore esegue una versione LTS:

1. Verifica di utilizzare la versione 5.0.0 del motore utilizzando: AWS CLI

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

2. Se necessario, aggiorna la versione del cluster 5.0.0 del motore esistente. Per istruzioni su come eseguire un aggiornamento della patch, vedere. Esecuzione di un aggiornamento della patch alla versione del motore di un cluster

Supporto esteso Amazon DocumentDB

Amazon DocumentDB Extended Support consente di continuare a utilizzare una versione del motore oltre la data di supporto standard a un costo aggiuntivo. I tuoi cluster vengono iscritti al supporto esteso una volta trascorsa la fine della data di supporto standard. Se non completi gli aggiornamenti entro la data di supporto standard della versione del motore, i cluster vengono fatturati alla tariffa Extended Support.

Argomenti

- Panoramica del supporto esteso
- Costi del supporto esteso
- Responsabilità

Amazon DocumentDB Extended Support fornisce i seguenti aggiornamenti e supporto tecnico:

- Aggiornamenti di sicurezza per CVE critici e elevati per la tua istanza o cluster
- Correzioni di bug e patch per problemi critici
- La possibilità di aprire casi di supporto e ricevere assistenza per la risoluzione dei problemi nell'ambito del contratto di servizio standard di Amazon DocumentDB

Questa offerta a pagamento ti dà più tempo per passare a una versione del motore supportata. Ad esempio, la data di fine del supporto standard di Amazon DocumentDB per la versione 3.6 è il 30 marzo 2026, tuttavia non sei pronto per l'aggiornamento alla versione 5.0 o 8.0 prima di tale data. In questo caso, Amazon DocumentDB registra il cluster della versione 3.6 in Extended Support il 31 marzo 2026 in modo da poter continuare a utilizzare Amazon DocumentDB versione 3.6. A partire dal 1° luglio 2026, ti AWS addebiterà un premio di supporto esteso per i cluster Amazon DocumentDB versione 3.6.

Il supporto esteso di Amazon DocumentDB è disponibile per un massimo di 3 anni dopo la data di supporto standard della versione del motore. Trascorso questo periodo, se non hai aggiornato la versione del motore a una versione supportata, Amazon DocumentDB aggiornerà la versione del motore. Esegui l'upgrade a una versione del motore supportata il prima possibile.

Data di fine del supporto standard della versione 3.6 di Amazon DocumentDB e date di supporto esteso:

- Rilascio della versione 3.6 — 9 gennaio 2019
- Fine del supporto standard per la versione 3.6 — 30 marzo 2026
- Inizio della versione 3.6 Extended Support, prezzi per l'anno 1: 1 luglio 2026
- Inizio della versione 3.6 Extended Support, prezzi del terzo anno — 31 marzo 2028
- Fine della versione 3.6 Extended Support — 30 marzo 2029

Panoramica del supporto esteso

Amazon DocumentDB aggiorna il cluster all'ultima versione del motore rilasciata (prima della fine della data di supporto esteso), se il cluster non è già in quella versione. Il periodo di aggiornamento avverrà solo dopo la data di supporto standard per quella versione del motore.

Puoi creare nuovi cluster con una versione del motore che ha raggiunto la fine della data di supporto standard, se disponi già di un cluster su quella versione del motore. Amazon DocumentDB registra questi nuovi cluster in Amazon DocumentDB Extended Support e ti addebita questa offerta. Se esegui l'upgrade a una versione del motore che è ancora soggetta al supporto standard di Amazon DocumentDB prima della data di fine del supporto standard della versione 3.6, non ti verrà addebitato alcun premio per il supporto esteso.

Puoi terminare l'iscrizione ad Amazon DocumentDB Extended Support in qualsiasi momento. Per terminare l'iscrizione, aggiorna ogni cluster registrato a una versione più recente del motore che è ancora soggetta al supporto standard di Amazon DocumentDB. La fine dell'iscrizione ad Amazon DocumentDB Extended Support sarà effettiva il giorno in cui completerai l'aggiornamento a una versione più recente del motore che è ancora soggetta al supporto standard di Amazon DocumentDB.

Per ulteriori informazioni sulle date di fine del supporto standard di Amazon DocumentDB e sulle date di fine del supporto esteso di Amazon DocumentDB, consulta. Date di supporto delle versioni del motore Amazon DocumentDB

Costi del supporto esteso

Ti verranno addebitati i costi per tutti i cluster in esecuzione su Extended Support a partire dal 1° luglio 2026. Per la data di fine del supporto standard della versione 3.6 di Amazon DocumentDB, consulta. Date di supporto delle versioni del motore Amazon DocumentDB

Il costo aggiuntivo per Amazon DocumentDB Extended Support si interrompe quando esegui una delle seguenti azioni:

- Esegui l'upgrade a una versione del motore coperta dal supporto standard
- Elimina il cluster su cui è in esecuzione la versione 3.6 dopo la data di fine del supporto standard

Ad esempio, la versione 3.6 di Amazon DocumentDB entra in Extended Support il 31 marzo 2026, ma i costi non iniziano fino al 1° luglio 2026. Se aggiorni il tuo cluster Amazon DocumentDB versione 3.6 alla versione 5.0 di Amazon DocumentDB il 31 luglio 2026, ti verranno addebitati solo 30 giorni di supporto esteso su Amazon DocumentDB versione 3.6. Per ulteriori informazioni, consulta i prezzi di Amazon DocumentDB. https://aws.amazon.com/documentdb/pricing/

Responsabilità

Di seguito sono elencate le responsabilità di Amazon DocumentDB e le tue responsabilità con Amazon DocumentDB Extended Support.

Responsabilità di Amazon DocumentDB

Dopo la fine della data di supporto standard della versione 3.6 di Amazon DocumentDB, Amazon DocumentDB applicherà patch, correzioni di bug e aggiornamenti per i cluster della versione 3.6 registrati in Extended Support.

Le tue responsabilità

Sei responsabile dell'aggiornamento del motore a una versione più recente prima della data di fine del supporto esteso della versione 3.6 di Amazon DocumentDB. Se non aggiorni il cluster, dopo la data di scadenza del supporto esteso della versione 3.6 di Amazon DocumentDB, Amazon DocumentDB tenterà di aggiornare il cluster alla versione più recente del motore supportata dal supporto standard di Amazon DocumentDB.

Caratteristiche e configurazioni di Amazon DocumentDB

Le tabelle seguenti confrontano le caratteristiche, le funzionalità e i tipi di istanza supportati tra le versioni del motore Amazon DocumentDB e i cluster elastici. Utilizza queste tabelle per determinare quale versione del motore si adatta meglio ai requisiti del carico di lavoro.

Stato del motore

Stato del motore per versione

Caratteristica/capacità	v3.6	v4.0	v5.0	v8.0	
Stato del supporto	Data di scadenza: marzo 2026	Attivo	Attivo (LTS disponibile)	Attivo (GA, novembre 2025)	
Supporto esteso disponibile	Sì (a pagamento, fino a marzo 2029)	N/A	N/A	N/A	
Versione LTS disponibile	No	No	Sì (v5.0 LTS)	No	
Cluster elastici	No	No	Sì	No	

Tipi di cluster

Tipi di cluster per versione

Cluster Type (Tipo di cluster)	v3.6	v4.0	v5.0	v8.0	Cluster elastici
Instance-based grappolo	Sì	Sì	Sì	Sì	Sì
istanze serverless	No	No	Sì	Sì	No

Funzionalità

Caratteristiche e configurazioni per versione del motore

Funzionalità	v3.6	v4.0	v5.0	v8.0	Cluster elastici
Cluster globali	No	Sì	Sì	Sì	No
Multi-AZ (3 AZ)	Sì	Sì	Sì	Sì	Sì (per frammento)
Repliche di lettura supportate	15	15	15	15	15 per frammento
Transazioni ACID	No	Sì	Sì	Sì	No
Scrive riutilizzabili	No	No	No	Sì (8.0.2+)	No
Change streams	Sì	Sì	Sì	Sì	No
Ricerca vettoriale	No	No	Sì	Sì	No
Approfondimenti sulle prestazioni	Sì	Sì	Sì	Sì	No
In-place aggiornamento della versione principale (MVU)	Sì (a 5.0 o 8.0)	Sì (fino a 5.0 o 8.0)	Sì (fino a 8.0)	No (solo bersaglio)	No
Client-side crittografia a livello di campo (FLE)	No	No	Sì	Sì	No
I/O-Optimized archiviazione	No	No	Sì	Sì	No
rotazione del certificato TLS (nessun riavvio)	Sì (patch 1.0.208662+)	Sì (patch 2.0.10179+)	Sì (patch 3.0.4780+)	Sì	Sì

Funzionalità	v3.6	v4.0	v5.0	v8.0	Cluster elastici
Cross-Region copia istantanea	Sì	Sì	Sì	Sì	Sì
Cross-account condivisione di istantanee	Sì	Sì	Sì	Sì	No
Log di controllo e di interrogazione lenta	Sì	Sì	Sì	Sì	Sì
Dual-stack IPv4/IPv6	No	Sì (patch 2.0.11747 +)	Sì (patch 3.0.15902 +)	Sì	N/A
Collation (Regola di confronto)	No	No	No	Sì	No
Visualizzazioni	No	No	No	Sì	No
Supporto della compressione	No	No	LZ4	LZ4/Ztsd	No
Indice di testo	No	No	V1	V2	No
FedRAMP () GovCloud	No	No	Sì	Sì	No

Tipi di istanza

Supporto del tipo di istanza per versione

Tipo di istanza	v3.6	v4.0	v5.0	v8.0	
db.r4 (generazione precedente)	Sì	No	No	No	
db.r5	Sì	Sì	Sì	Sì	

Tipo di istanza	v3.6	v4.0	v5.0	v8.0	
db.r6g (Graviton2)	No	Sì	Sì	Sì	
db.r6gd (NVMe-backed Gravitone2)	No	No	Sì	Sì	
db.r8g (Gravitone4)	No	No	Sì	Sì	
db.t3.medium	Sì	Sì	Sì	Sì	
db.t4g.medium (Graviton 2)	No	Sì	Sì	Sì	
serverless	No	No	Sì	Sì	

AWS integrazioni

AWS supporto all'integrazione per versione

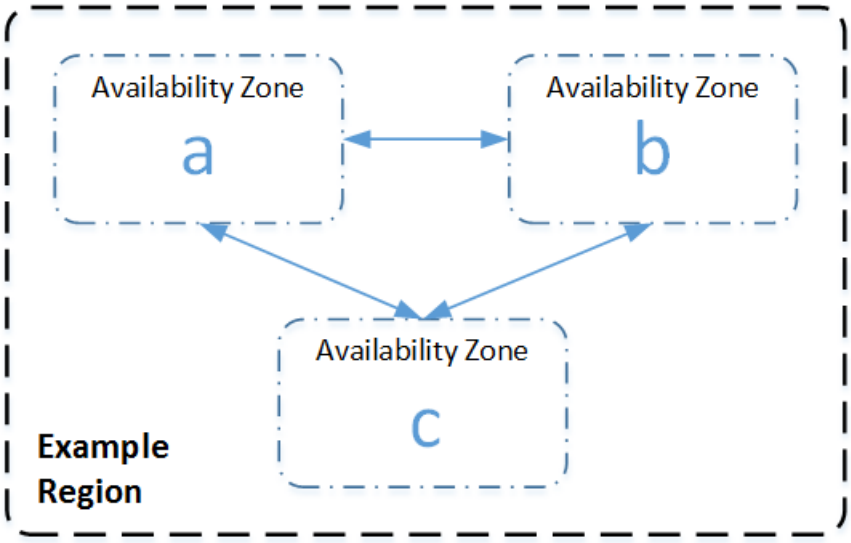
AWS Integrazione/funzionalità	v3.6	v4.0	v5.0	v8.0	Cluster elastici
AWS Backup	Sì	Sì	Sì	Sì	No
AWS Secrets Manager	Sì	Sì	Sì	Sì	No
AWS Lambda (ESM nativo)	No	Sì	Sì	No	No
Amazon OpenSearch Service Ingestion () Zero-ETL	No	Sì	Sì	Sì	No
AWS DMS supporto per la migrazione	Sì	Sì	Sì	Sì (da 5.0 in poi AWS DMS)	Sì

AWS Integrazione/funzionalità	v3.6	v4.0	v5.0	v8.0	Cluster elastici
AWS Identity and Access Management authentication	No	No	Sì (basato sull'istanza 5.0)	Sì	No
AWS CloudShell	No	No	Sì	Sì	No
AWS Key Management Service	Sì	Sì	Sì	Sì	Sì
AWS CloudTrail	Sì	Sì	Sì	Sì	Sì

Scelta di regioni e zone di disponibilità

Le risorse di cloud computing Amazon sono ospitate in più ubicazioni in tutto il mondo. Queste sedi sono costituite da zone di Regioni AWS disponibilità. Ciascuna Regione AWSè un'area geografica separata. Ciascuna regione presenta più località isolate, conosciute come zone di disponibilità. Amazon DocumentDB ti offre la possibilità di collocare risorse, come istanze e dati, in più posizioni. Le risorse non vengono replicate tra di loro Regioni AWS a meno che tu non lo faccia in modo specifico.

Amazon gestisce data center avanzati e ad alta disponibilità. Anche se rari, i guasti che compromettono la disponibilità di istanze nella stessa ubicazione possono verificarsi. Se ospiti tutte le istanze in un'unica ubicazione colpita da tale guasto, nessuna di esse risulterà disponibile. Il diagramma seguente mostra una Regione AWS con tre zone di disponibilità.



È importante ricordare che ogni regione è indipendente. Qualsiasi attività di Amazon DocumentDB avviata (ad esempio, la creazione di istanze o l'elenco delle istanze disponibili) viene eseguita solo nell'impostazione predefinita corrente. Regione AWS Puoi modificare la regione predefinita nella console impostando la variabile di ambiente `EC2_REGION`. In alternativa, puoi sostituirla utilizzando il parametro `--region` nell' AWS CLI. Per ulteriori informazioni, consulta Configurazione, in particolare AWS Command Line Interface, delle sezioni sulle variabili di ambiente e sulle opzioni della riga di comando.

Quando crei un cluster utilizzando la console Amazon DocumentDB e scegli di creare una replica in una zona di disponibilità diversa, Amazon DocumentDB crea due istanze. Crea l'istanza primaria in una zona di disponibilità e l'istanza di replica in un'altra zona di disponibilità. Il volume cluster viene sempre replicato nelle tre zone di disponibilità.

Per creare o utilizzare un'istanza Amazon DocumentDB in una specifica istanza Regione AWS, utilizza l'endpoint di servizio regionale corrispondente.

Disponibilità nelle regioni

Amazon DocumentDB è disponibile nelle seguenti AWS regioni.

Regioni supportate da Amazon DocumentDB

Nome della regione	Regione	Zone di disponibilità (elaborazione)
Stati Uniti orientali (Ohio)	`us-east-2`	3

Nome della regione	Regione	Zone di disponibilità (elaborazione)
Stati Uniti orientali (Virginia settentrionale)	us-east-1	6
Stati Uniti occidentali (Oregon)	us-west-2	4
Africa (Città del Capo)	af-south-1	3
Sud America (San Paolo)	sa-east-1	3
Asia Pacifico (Hong Kong)	ap-east-1	3
Asia Pacifico (Hyderabad)	ap-south-2	3
Asia Pacifico (Malesia)	ap-southeast-5	3
Asia Pacifico (Mumbai)	ap-south-1	3
Asia Pacifico (Osaka)	ap-northeast-3	3
Asia Pacifico (Seul)	ap-northeast-2	4
Asia Pacifico (Singapore)	ap-southeast-1	3
Asia Pacifico (Sydney)	ap-southeast-2	3
Asia Pacifico (Giacarta)	ap-southeast-3	3

Nome della regione	Regione	Zone di disponibilità (elaborazione)
Asia Pacifico (Melbourne)	ap-southeast-4	3
Asia Pacifico (Thailandia)	ap-southeast-7	3
Asia Pacifico (Tokyo)	ap-northeast-1	3
Canada (Centrale)	ca-central-1	3
Canada occidentale (Calgary)	ca-west-1	3
Regione Cina (Pechino)	cn-north-1	3
Cina (Ningxia)	cn-northwest-1	3
Europa (Francoforte)	eu-central-1	3
Europa (Zurigo)	eu-central-2	3
Europa (Irlanda)	eu-west-1	3
Europa (Londra)	eu-west-2	3
Europa (Milano)	eu-south-1	3
Europa (Parigi)	eu-west-3	3
Europa (Spagna)	eu-south-2	3
Europa (Stoccolma)	eu-north-1	3
Messico (centrale)	mx-central-1	3
Medio Oriente (Emirati Arabi Uniti)	me-central-1	3

Nome della regione	Regione	Zone di disponibilità (elaborazione)
Israele (Tel Aviv)	`il-central-1`	3
AWS GovCloud (US-West)	`us-gov-west-1`	3
AWS GovCloud (US-East)	`us-gov-east-1`	3

Per impostazione predefinita, il fuso orario per un cluster Amazon DocumentDB è Universal Time Coordinated (UTC).

Per informazioni su come trovare gli endpoint di connessione per cluster e istanze in una particolare regione, consulta. Comprendere gli endpoint di Amazon DocumentDB

Zero-ETL integrazione con Amazon OpenSearch Service

Argomenti

- Amazon OpenSearch Service come destinazione
- Limitazioni

Amazon OpenSearch Service come destinazione

OpenSearch L'integrazione del servizio con Amazon DocumentDB consente di trasmettere eventi di dati a pieno carico e di modificare i OpenSearch domini. L'infrastruttura di ingestion è ospitata come pipeline di OpenSearch ingestione e fornisce un meccanismo su larga scala e bassa latenza per lo streaming continuo di dati dalle raccolte Amazon DocumentDB.

Durante il pieno carico, l'integrazione Zero-ETL estrae innanzitutto i dati storici a pieno carico utilizzando una pipeline di ingestione. OpenSearch Una volta inseriti i dati a pieno carico, le pipeline di OpenSearch ingestione inizieranno a leggere i dati provenienti dai flussi di modifica di Amazon DocumentDB e alla fine recupereranno il ritardo per mantenere la coerenza dei dati quasi in tempo reale tra Amazon DocumentDB e. OpenSearch OpenSearch archivia i documenti in indici. I dati in entrata da una raccolta Amazon DocumentDB possono essere inviati a un indice o possono essere partizionati in indici diversi. Le pipeline di ingestione sincronizzeranno tutti gli eventi di creazione,

aggiornamento ed eliminazione in una raccolta Amazon DocumentDB come corrispondente creazione, aggiornamento ed eliminazione dei documenti per mantenere sincronizzati entrambi i sistemi di OpenSearch dati. Le pipeline di ingestione possono essere configurate per leggere i dati da una raccolta e scrivere su un indice o leggere i dati da una raccolta e indirizzarli condizionatamente a più indici.

Le pipeline di ingestione possono essere configurate per lo streaming di dati da Amazon DocumentDB ad Amazon Service utilizzando: OpenSearch

- Solo a pieno carico
- Trasmetti in streaming gli eventi di change stream da Amazon DocumentDB senza caricamento completo
- Caricamento completo seguito dai flussi di modifiche da Amazon DocumentDB

Per configurare la pipeline di ingestione, procedi nel seguente modo:

Passaggio 1: crea un dominio Amazon OpenSearch Service o una raccolta serverless OpenSearch

È necessaria una raccolta Amazon OpenSearch Service con le autorizzazioni appropriate per leggere i dati. Per creare una raccolta, consulta Getting started with Amazon OpenSearch Service o Getting started with Amazon OpenSearch Serverless nella Amazon OpenSearch Service Developer Guide. Fai riferimento ad Amazon OpenSearch Ingestion nella Amazon OpenSearch Service Developer Guide per creare un ruolo AIM con le autorizzazioni corrette per accedere ai dati di scrittura nella raccolta o nel dominio.

Passaggio 2: abilitare i flussi di modifica sul cluster Amazon DocumentDB

Assicurati che i flussi di modifica siano abilitati sulle raccolte richieste nel cluster Amazon DocumentDB. Per ulteriori informazioni, consulta Utilizzo dei flussi di modifiche con Amazon DocumentDB.

Passaggio 3: configura il ruolo della pipeline con le autorizzazioni di scrittura nel bucket Amazon S3 e nel dominio o nella raccolta di destinazione

Dopo aver creato la raccolta Amazon DocumentDB e aver abilitato il change stream, configura il ruolo della pipeline che desideri utilizzare nella configurazione della pipeline e aggiungi le seguenti autorizzazioni nel ruolo:

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/*"
            ]
        }
    ]
}
```

Affinché una OpenSearch pipeline possa scrivere dati su un OpenSearch dominio, il dominio deve disporre di una policy di accesso a livello di dominio che consenta al ruolo della pipeline sts_role_arn di accedervi. Il seguente esempio di politica di accesso al dominio consente al ruolo della pipeline denominato`pipeline-role`, creato nel passaggio precedente, di scrivere dati nel dominio denominato: `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}/*"
    }
  ]
}
```

Passaggio 4: aggiungere le autorizzazioni richieste sul ruolo della pipeline da creare 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" }
            }
        }
    ]
}
```

Passaggio 5: creare la pipeline

Configura una pipeline OpenSearch di ingestion specificando Amazon DocumentDB come fonte. Questa configurazione di pipeline di esempio presuppone l'uso di un meccanismo di recupero del flusso di modifiche. Per ulteriori informazioni, consulta Using an OpenSearch Ingestion pipeline with Amazon DocumentDB nella Amazon Service Developer Guide. OpenSearch

Limitazioni

Le seguenti limitazioni si applicano all'integrazione di Amazon DocumentDB: OpenSearch

- È supportata una sola raccolta Amazon DocumentDB come origine per pipeline.
- Cross-region l'inserimento di dati non è supportato. Il cluster e il OpenSearch dominio Amazon DocumentDB devono appartenere allo stesso. Regione AWS
- Cross-account l'inserimento di dati non è supportato. Il cluster Amazon DocumentDB e la pipeline OpenSearch di ingestione devono trovarsi nello stesso account. AWS
- I cluster elastici Amazon DocumentDB non sono supportati. Sono supportati solo i cluster basati su istanze Amazon DocumentDB.
- Assicurati che il cluster Amazon DocumentDB abbia l'autenticazione abilitata utilizzando i segreti. AWS AWS i segreti sono l'unico meccanismo di autenticazione supportato.
- La configurazione della pipeline esistente non può essere aggiornata per importare dati da un database diverso da and/or una raccolta diversa. Per aggiornare il nome della and/or raccolta del database di una pipeline, è necessario creare una nuova pipeline.

Sviluppo con Amazon DocumentDB

Questa sezione riguarda la connessione al cluster Amazon DocumentDB, l'interrogazione e la gestione dei dati, l'utilizzo degli indici e l'utilizzo di funzionalità come flussi di modifiche, transazioni e ricerca vettoriale per applicazioni di intelligenza artificiale generativa.

Argomenti

- Connect ad Amazon DocumentDB
- Programma con Amazon DocumentDB
- Interrogazione in Amazon DocumentDB
- Indici
- Utilizzo dei flussi di modifiche con Amazon DocumentDB
- Transazioni in Amazon DocumentDB
- Retryable scrive in Amazon DocumentDB
- Utilizzo delle regole di confronto in Amazon DocumentDB 8.0
- Utilizzo delle viste in Amazon DocumentDB 8.0
- Intelligenza artificiale generativa di Amazon DocumentDB
- Utilizzo AWS Lambda con flussi di modifica

Connect ad Amazon DocumentDB

Amazon DocumentDB offre diverse opzioni di connessione, tra cui connessioni programmatiche che utilizzano driver MongoDB, connessioni dall'esterno di un VPC tramite tunneling SSH o VPN e strumenti popolari come Studio 3T e. AWS DataGrip Il servizio supporta connessioni con set di repliche per un'elevata disponibilità. Per l'analisi e il reporting, puoi connetterti utilizzando i driver JDBC o ODBC. Le istanze EC2 nello stesso VPC possono connettersi direttamente ai cluster DocumentDB. Tutte le connessioni richiedono TLS e appropriate. AWS credentials/authentication

Argomenti

- Connessione programmatica ad Amazon DocumentDB
- Connessione a un cluster Amazon DocumentDB dall'esterno di Amazon VPC
- Connessione ad Amazon DocumentDB come set di repliche

- Connessione a un cluster Amazon DocumentDB da Studio 3T
- Connect ad Amazon DocumentDB tramite DataGrip
- Connettiti tramite Amazon EC2
- Connettiti utilizzando il driver JDBC di Amazon DocumentDB
- Connettiti utilizzando il driver ODBC di Amazon DocumentDB

Connessione programmatica ad Amazon DocumentDB

Questa sezione contiene esempi di codice che dimostrano come connettersi ad Amazon DocumentDB (con compatibilità MongoDB) utilizzando diversi linguaggi. Gli esempi sono suddivisi in due sezioni diverse, a seconda che sul cluster con cui ti connetti sia abilitato o meno il protocollo Transport Layer Security (TLS). Per impostazione predefinita, TLS è abilitato sui cluster Amazon DocumentDB. è comunque possibile disattivarlo, se necessario. Per ulteriori informazioni, consulta Crittografia dei dati in transito.

Se stai tentando di connetterti ad Amazon DocumentDB dall'esterno del VPC in cui risiede il cluster, consulta. Connessione a un cluster Amazon DocumentDB dall'esterno di Amazon VPC

Prima di connetterti al cluster, devi sapere se sul cluster è abilitato il protocollo TLS. La sezione successiva illustra come determinare il valore del parametro `tls` del cluster tramite la Console di gestione AWS o l' AWS CLI. Seguendo tale procedura potrai trovare e applicare il codice di esempio appropriato.

Argomenti

- Determinazione del valore del parametro tls
- Connessione con TLS abilitato
- Connessione con TLS disabilitato
- Compressione di rete in Amazon DocumentDB

Determinazione del valore del parametro tls

Determinare se il TLS nel cluster è abilitato è un processo in due fasi che puoi eseguire utilizzando o. Console di gestione AWS AWS CLI

1. Determinare il gruppo di parametri che regola il cluster.

Using the Console di gestione AWS

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Nell'elenco dei cluster selezionare il nome del cluster.
4. La pagina risultante mostra i dettagli del cluster selezionato. Selezionare la scheda Configurazione. Nella sezione Configurazioni e stato, individua il nome del gruppo di parametri sotto il gruppo di parametri Cluster.

Using the AWS CLI

Il AWS CLI codice seguente determina il parametro che governa il cluster. Assicurarsi di sostituire *sample-cluster* con il nome del cluster.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
[
       [
           "sample-cluster",
           "sample-parameter-group"
       ]
]
```

2. Determinare il valore del parametro **tls** nel gruppo di parametri del cluster.

Using the Console di gestione AWS

1. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
2. Nella finestra Gruppi di parametri del cluster, seleziona il nome del gruppo di parametri del cluster dal passaggio 1d.

3. La pagina risultante mostra i parametri del gruppo di parametri del cluster. Puoi vedere il valore del parametro `tls` qui. Per informazioni sulla modifica di questo parametro, consulta Modifica dei gruppi di parametri del cluster Amazon DocumentDB.

Using the AWS CLI

È possibile utilizzare il `describe-db-cluster-parameters` AWS CLI comando per visualizzare i dettagli dei parametri nel gruppo di parametri del cluster.

- **`--describe-db-cluster-parameters`**— Per elencare tutti i parametri all'interno di un gruppo di parametri e i relativi valori.
  - **`--db-cluster-parameter-group name`**: obbligatorio. Il nome del gruppo di parametri del cluster.

Negli esempi seguenti, sostituisci ciascuno *`user input placeholder`* con le informazioni del tuo cluster.

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
        "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 supporta endpoint FIPS 140-3 a partire dai cluster Amazon DocumentDB 5.0 (versione del motore 3.0.3727) nelle seguenti regioni: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1.

Dopo aver determinato il valore del parametro `tls`, continuare con la connessione al cluster utilizzando uno degli esempi di codice nelle seguenti sezioni.

- Connessione con TLS abilitato
- Connessione con TLS disabilitato

Connessione con TLS abilitato

Per visualizzare un esempio di codice per la connessione programmatica a un cluster TLS-enabled Amazon DocumentDB, scegli la scheda appropriata per la lingua che desideri utilizzare.

Per crittografare i dati in transito, scarica la chiave pubblica per Amazon DocumentDB denominata `global-bundle.pem` utilizzando la seguente operazione.

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

Se l'applicazione è su Microsoft Windows e richiede un file PKCS7, puoi scaricare il bundle di certificati PKCS7. Questo pacchetto contiene sia i certificati intermedi che quelli principali all'indirizzo. https://truststore.pki.rds.amazonaws.com/global/global-bundle.p7b

Python

Il codice seguente dimostra come connettersi ad Amazon DocumentDB utilizzando Python quando TLS è abilitato.

Nel seguente esempio, sostituisci ciascuno *user input placeholder* con le informazioni del tuo cluster.

```
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

Il codice seguente dimostra come connettersi ad Amazon DocumentDB utilizzando Node.js quando TLS è abilitato.

> **Important**
>
> Esiste una limitazione nota con Node.js i driver precedenti alla versione 6.13.1, che attualmente non sono supportati dall'autenticazione dell'identità IAM per Amazon DocumentDB. Node.js i driver e gli strumenti che utilizzano Node.js driver (ad esempio, mongosh) devono essere aggiornati per utilizzare la versione 6.13.1 o successiva. Node.js

Nell'esempio seguente, sostituisci ciascuno *user input placeholder* con le informazioni del tuo cluster.

```
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

Il codice seguente dimostra come connettersi ad Amazon DocumentDB utilizzando PHP quando TLS è abilitato.

Nel seguente esempio, sostituisci ciascuno *user input placeholder* con le informazioni del tuo cluster.

```
<?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

Il codice seguente dimostra come connettersi ad Amazon DocumentDB utilizzando Go quando TLS è abilitato.

> **Note**
>
> A partire dalla versione 1.2.1, il driver MongoDB Go utilizzerà solo il primo certificato del server CA trovato in `sslcertificateauthorityfile`. Il codice di esempio riportato di seguito risolve questa limitazione aggiungendo manualmente tutti i certificati server trovati in `sslcertificateauthorityfile` su una configurazione TLS personalizzata utilizzata durante la creazione del client.

Nei seguenti esempi, sostituisci ciascuno *`user input placeholder`* con le informazioni del tuo cluster.

```
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

Quando ci si connette a un cluster TLS-enabled Amazon DocumentDB da un'applicazione Java, il programma deve utilizzare il file AWS di autorità di certificazione (CA) fornito per convalidare la connessione. Per utilizzare il certificato Amazon RDS CA, procedi come segue:

1. Scarica il file Amazon RDS CA da. https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
2. Creare uno store attendibile con il certificato CA contenuto nel file eseguendo i comandi indicati di seguito. Assicurati di cambiarlo con *truststore-password* qualcos'altro. Se si accede a uno store attendibile che contiene sia il vecchio certificato CA (`rds-ca-2015-root.pem`) sia il nuovo certificato CA (`rds-ca-2019-root.pem`), è possibile importare il pacchetto di certificati nell'archivio attendibilità.

   Il seguente script è uno script di esempio shell che importa il bundle di certificati in un archivio di trust su un sistema operativo Linux. Negli esempi seguenti, sostituisci ciascuno *user input placeholder* con le tue informazioni. In particolare, ovunque la directory di esempio "*mydir*" si trovi nello script, sostituiscila con una directory creata per questa attività.

   ```
   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
```

Il seguente script è uno script di shell di esempio che importa il bundle di certificati in un archivio di trust su un sistema operativo Linux.

```
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. Utilizzalo `keystore` nel tuo programma impostando le seguenti proprietà di sistema nell'applicazione prima di effettuare una connessione al cluster Amazon DocumentDB.

```
javax.net.ssl.trustStore: truststore
javax.net.ssl.trustStorePassword: truststore-password;
```

4. Il codice seguente dimostra come connettersi ad Amazon DocumentDB utilizzando Java quando TLS è abilitato.

   Nel seguente esempio, sostituisci ciascuno *user input placeholder* con le informazioni del tuo cluster.

```
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

Il codice seguente dimostra come connettersi ad Amazon DocumentDB utilizzando C#/.NET quando TLS è abilitato.

Nel seguente esempio, sostituisci ciascuno *`user input placeholder`* con le informazioni del tuo cluster.

```
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

Il codice seguente dimostra come connettersi e interrogare Amazon DocumentDB utilizzando la versione più recente, mongosh, o la versione precedente di mongo shell, quando TLS è abilitato.

Connettiti ad Amazon DocumentDB con mongosh

> **Important**
>
> Esiste una limitazione nota con Node.js i driver precedenti alla versione 6.13.1, che attualmente non sono supportati dall'autenticazione dell'identità IAM per Amazon DocumentDB. Node.js i driver e gli strumenti che utilizzano Node.js driver (ad esempio, mongosh) devono essere aggiornati per utilizzare la versione 6.13.1 o successiva. Node.js

Negli esempi seguenti, sostituisci ciascuno *`user input placeholder`* con le informazioni del tuo cluster.

```
mongosh --tls --host cluster-end-point:27017 --tlsCAFile  global-bundle.pem --
username sample-user --password password --retryWrites false
```

Connettiti ad Amazon DocumentDB con la precedente versione di mongo shell

Se usi IAM, devi usare una versione precedente di mongo shell. Inserisci una delle seguenti opzioni di comando:

```
mongo --ssl --host cluster-end-point:27017 --sslCAFile global-bundle.pem --
username sample-user --password password
```

Se stai usando una versione uguale o superiore alla 4.2, usa il codice seguente per connetterti. A partire dalla versione 8.0.2 del motore, Amazon DocumentDB supporta scritture riutilizzabili.

Nelle versioni precedenti del motore, le scritture riutilizzabili non sono supportate. Se stai usando la shell mongo legacy (non mongosh), non includere il comando in nessuna stringa di codice. `retryWrites=false` Per ulteriori informazioni, consulta Retryable scrive in Amazon DocumentDB.

```
mongo --tls --host cluster-end-point:27017 --tlsCAFile global-bundle.pem --
username sample-user --password password
```

Prova la connessione

1. Inserire un singolo documento.

   ```
   db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
   ```

2. Trovare il documento precedentemente inserito.

   ```
   db.myTestCollection.find({'hello':'Amazon DocumentDB'})
   ```

R

Il codice seguente dimostra come connettersi ad Amazon DocumentDB con R utilizzando mongolite (https://jeroen.github.io/mongolite/) quando TLS è abilitato.

Nel seguente esempio, sostituisci ciascuno *user input placeholder* con le informazioni del tuo cluster.

```
#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

Il codice seguente dimostra come connettersi ad Amazon DocumentDB con Ruby quando TLS è abilitato.

Nel seguente esempio, sostituisci ciascuno *user input placeholder* con le informazioni del tuo cluster.

```
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
```

Connessione con TLS disabilitato

Per visualizzare un esempio di codice per la connessione programmatica a un cluster TLS-disabled Amazon DocumentDB, scegli la scheda relativa alla lingua che desideri utilizzare.

Python

Il codice seguente dimostra come connettersi ad Amazon DocumentDB utilizzando Python quando TLS è disabilitato.

Nel seguente esempio, sostituisci ciascuno *user input placeholder* con le informazioni del tuo cluster.

```
## 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

Il codice seguente dimostra come connettersi ad Amazon DocumentDB utilizzando Node.js quando TLS è disabilitato.

> **Important**
>
> Esiste una limitazione nota con Node.js i driver precedenti alla versione 6.13.1, che attualmente non sono supportati dall'autenticazione dell'identità IAM per Amazon DocumentDB. Node.js i driver e gli strumenti che utilizzano Node.js driver (ad esempio, mongosh) devono essere aggiornati per utilizzare la versione 6.13.1 o successiva. Node.js

Nell'esempio seguente, sostituisci ciascuno *`user input placeholder`* con le informazioni del tuo cluster.

```
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

Il codice seguente dimostra come connettersi ad Amazon DocumentDB utilizzando PHP quando TLS è disabilitato.

Nel seguente esempio, sostituisci ciascuno *user input placeholder* con le informazioni del tuo cluster.

```
<?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

Il codice seguente dimostra come connettersi ad Amazon DocumentDB utilizzando Go quando TLS è disabilitato.

Nel seguente esempio, sostituisci ciascuno *user input placeholder* con le informazioni del tuo cluster.

```
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

Il codice seguente dimostra come connettersi ad Amazon DocumentDB utilizzando Java quando TLS è disabilitato.

Nel seguente esempio, sostituisci ciascuno *user input placeholder* con le informazioni del tuo cluster.

```
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

Il codice seguente dimostra come connettersi ad Amazon DocumentDB utilizzando C#/.NET quando TLS è disabilitato.

Nel seguente esempio, sostituisci ciascuno *user input placeholder* con le informazioni del tuo cluster.

```
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

Il codice seguente dimostra come connettersi e interrogare Amazon DocumentDB utilizzando la versione più recente, mongosh, o la versione precedente di mongo shell, quando TLS è disabilitato.

Connettiti ad Amazon DocumentDB con mongosh

> **Important**
>
> Esiste una limitazione nota con Node.js i driver precedenti alla versione 6.13.1, che attualmente non sono supportati dall'autenticazione dell'identità IAM per Amazon DocumentDB. Node.js i driver e gli strumenti che utilizzano Node.js driver (ad esempio, mongosh) devono essere aggiornati per utilizzare la versione 6.13.1 o successiva. Node.js

Negli esempi seguenti, sostituisci ciascuno *user input placeholder* con le informazioni del tuo cluster.

```
mongosh --host cluster-end-point:27017 --username sample-user --password password --
retryWrites false
```

Connettiti ad Amazon DocumentDB con la precedente versione di mongo shell

Se usi IAM, devi usare una versione precedente di mongo shell. Inserisci una delle seguenti opzioni di comando:

```
mongo --host cluster-end-point:27017 --username sample-user --password password
```

Se stai usando una versione uguale o superiore alla 4.2, usa il codice seguente per connetterti. Le scritture riutilizzabili non sono supportate in Amazon DocumentDB. Se utilizzi la shell mongo legacy (non mongosh), non includere il comando in nessuna stringa di codice. `retryWrites=false` Per impostazione predefinita, le scritture riutilizzabili sono disabilitate. L'inclusione `retryWrites=false` potrebbe causare un errore nei normali comandi di lettura.

```
mongo --host cluster-end-point:27017 --username sample-user --password password
```

Verifica la connessione

1. Inserire un singolo documento.

   ```
   db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
   ```

2. Trovare il documento precedentemente inserito.

   ```
   db.myTestCollection.find({'hello':'Amazon DocumentDB'})
   ```

R

Il codice seguente dimostra come connettersi ad Amazon DocumentDB con R utilizzando mongolite (https://jeroen.github.io/mongolite/) quando TLS è disabilitato.

Nel seguente esempio, sostituisci ciascuno *user input placeholder* con le informazioni del tuo cluster.

```
#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

Il codice seguente dimostra come connettersi ad Amazon DocumentDB con Ruby quando TLS è disabilitato.

Nel seguente esempio, sostituisci ciascuno *user input placeholder* con le informazioni del tuo cluster.

```
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
```

Compressione di rete in Amazon DocumentDB

Con la compressione tramite protocollo cablato, puoi ridurre i dati inviati dall'applicazione in rete. Amazon DocumentDB 8.0.1 e versioni successive supportano la compressione. Ottieni un throughput migliorato per i carichi di lavoro che trasferiscono documenti di grandi dimensioni o set di risultati di grandi dimensioni. Dopo aver abilitato la compressione, il driver e il server si scambiano messaggi compressi.

Quando si attiva la compressione di rete, si scambiano le risorse della CPU con una larghezza di banda di rete ridotta. Attiva la compressione quando il carico di lavoro è collegato alla rete e trasferisce payload di grandi dimensioni. Per richieste e risposte di piccole dimensioni, potresti non vedere un vantaggio netto, perché il sovraccarico di compressione può superare i risparmi.

Compressori supportati

Amazon DocumentDB supporta il `snappy` compressore. Amazon DocumentDB non supporta altri compressori offerti dai driver MongoDB, come e. `zlib` `zstd`

Se un driver richiede un elenco di compressori, Amazon DocumentDB negozia solo quelli supportati. Ad esempio, se un driver offre `snappy ezstd`, Amazon DocumentDB seleziona e ignora. `snappy zstd` Se un driver offre solo compressori non supportati, Amazon DocumentDB non comprime la connessione. In questo caso, il driver e il server si scambiano messaggi senza compressione.

Abilita la compressione

Per abilitare la compressione, aggiungi l'`compressors`opzione alla stringa di connessione e impostala su`snappy`. Nella seguente stringa di connessione *`usernamepassword`*, sostituisci e *`docdb-cluster-endpoint`* con i tuoi valori:

```
mongodb://username:password@docdb-cluster-endpoint:27017/?
tls=true&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false&compressors=snappy
```

Quando usi un driver MongoDB, imposta la stessa opzione tramite le opzioni di connessione del driver. La maggior parte dei driver richiede una dipendenza aggiuntiva sul lato client per eseguire una compressione rapida:

- Il driver richiede il pacchetto. Node.js `snappy`
- Il driver Python (PyMongo) richiede il `python-snappy` pacchetto. Su Linux, richiede `python-snappy` anche la libreria snappy di sistema (ad esempio, `snappy-devel` tramite `dnf` or). `yum`

Per il nome esatto dell'opzione e gli eventuali pacchetti richiesti, consultate la documentazione del driver.

> **Note**
>
> Se manca la dipendenza dalla compressione sul lato client, il driver si connette correttamente. Tuttavia, scambia messaggi senza compressione e non restituisce un errore. Per confermare che la compressione è attiva, verificate il compressore negoziato come descritto in. Verifica la compressione

Il seguente esempio di codice Python si connette al cluster con la compressione snappy abilitata e utilizza un `except` blocco `try` and per gestire gli errori di connessione:

```
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}')
```

Verifica la compressione

Per confermare che un driver e Amazon DocumentDB hanno negoziato la compressione, controlla il `compression` campo restituito dal comando `hello` MongoDB wire protocol. Quando il driver e Amazon DocumentDB negoziano la compressione, questo campo elenca i compressori concordati.

Connessione a un cluster Amazon DocumentDB dall'esterno di Amazon VPC

I cluster Amazon DocumentDB (con compatibilità MongoDB) vengono distribuiti all'interno di un Amazon Virtual Private Cloud (Amazon VPC). È possibile accedervi direttamente dalle istanze Amazon EC2 o da altri AWS servizi distribuiti nello stesso Amazon VPC. Inoltre, è possibile accedere ad Amazon DocumentDB da istanze EC2 o altri AWS servizi in diversi VPC nella stessa o in altre regioni tramite peering VPC. Regione AWS

Tuttavia, supponiamo che il tuo caso d'uso richieda che tu (o la tua applicazione) accedete alle risorse di Amazon DocumentDB dall'esterno del VPC del cluster. In tal caso, puoi utilizzare il tunneling SSH (noto anche come port forwarding) per accedere alle tue risorse Amazon DocumentDB.

La discussione approfondita sul tunneling SSH non rientra negli scopi di questo argomento. Per ulteriori informazioni sul tunneling SSH, consulta quanto segue:

- Tunnel SSH
- L'esempio sull'inoltro alla porta SSH e, in particolare, la sezione sull'inoltro locale

Per creare un tunnel SSH, è necessaria un'istanza Amazon EC2 in esecuzione nello stesso Amazon VPC del cluster Amazon DocumentDB. Puoi utilizzare un'istanza EC2 esistente nello stesso VPC del cluster o crearne una. Per ulteriori informazioni, consulta l'argomento rilevante per il tuo sistema operativo:

- Guida introduttiva alle istanze Linux di Amazon EC2
- Guida introduttiva alle istanze Windows di Amazon EC2

In genere, puoi connetterti a un'istanza EC2 con il comando seguente:

```
ssh -i "ec2Access.pem" ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com
```

In tal caso, puoi configurare un tunnel SSH verso il cluster Amazon DocumentDB `sample-cluster.node.us-east-1.docdb.amazonaws.com` eseguendo il seguente comando sul tuo computer locale. Il flag `-L` viene utilizzato per inoltrare una porta locale. Quando usi un tunnel SSH, connettiti al cluster utilizzando l'endpoint del cluster e non tentare di connetterti in modalità replica set (ad esempio, specificando `replicaSet=rs0` nella stringa di connessione) poiché si verificherà un errore.

```
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
```

Dopo la creazione del tunnel SSH, tutti i comandi inviati `localhost:27017` vengono inoltrati al cluster Amazon DocumentDB in esecuzione in Amazon VPC. `sample-cluster` Se Transport Layer Security (TLS) è abilitato sul tuo cluster Amazon DocumentDB, devi scaricare la chiave pubblica per Amazon DocumentDB da. https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem La seguente operazione scarica questo file:

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

> **Note**
>
> TLS è abilitato di default per i nuovi cluster Amazon DocumentDB. Tuttavia, è possibile effettuarne la disabilitazione. Per ulteriori informazioni, consulta Gestione delle impostazioni TLS del cluster Amazon DocumentDB.

Per connetterti al tuo cluster Amazon DocumentDB dall'esterno di Amazon VPC, utilizza il seguente comando.

```
mongo --sslAllowInvalidHostnames --ssl --sslCAFile global-bundle.pem --username
 <yourUsername> --password <yourPassword>
```

Connessione ad Amazon DocumentDB come set di repliche

Quando sviluppi con Amazon DocumentDB, connettiti al tuo cluster come set di repliche e distribuisci le letture alle istanze di replica utilizzando le funzionalità di preferenza di lettura integrate del tuo driver. Questa sezione approfondisce cosa significa e descrive come puoi connetterti al tuo cluster Amazon DocumentDB come set di repliche usando Python come esempio.

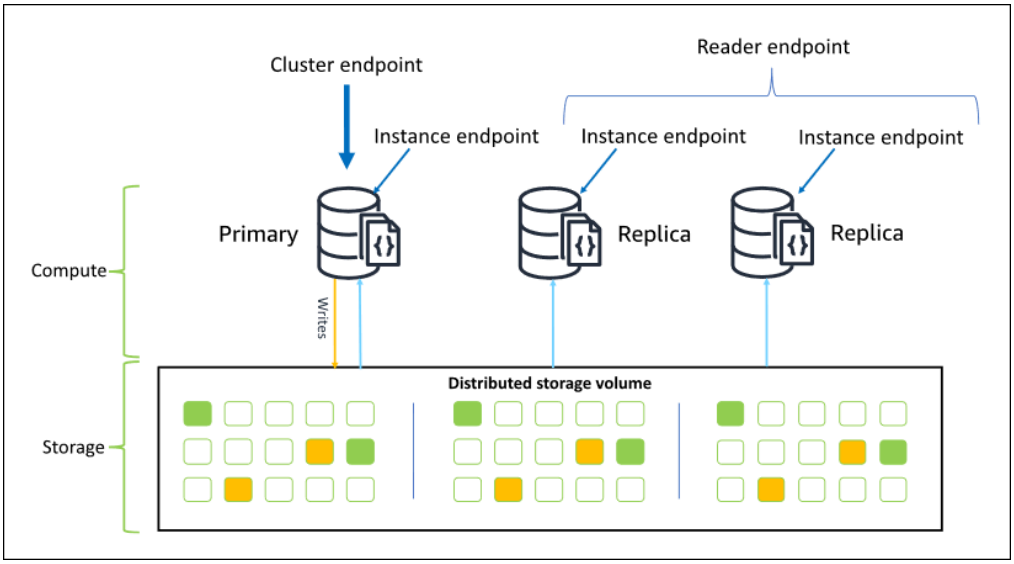
Amazon DocumentDB ha tre endpoint che puoi utilizzare per connetterti al tuo cluster:

- Endpoint del cluster
- Endpoint di lettura
- Endpoint dell'istanza

Quando ti connetti ad Amazon DocumentDB, usa l'endpoint del cluster. Si tratta di un CNAME che punta all'istanza primaria nel cluster, come mostrato nel seguente diagramma.

Quando usi un tunnel SSH, connettiti al cluster utilizzando l'endpoint del cluster. Non connettetevi in modalità set di repliche (specificando `replicaSet=rs0` nella stringa di connessione) poiché si verificherà un errore.

> **Note**
>
> Per ulteriori informazioni sugli endpoint di Amazon DocumentDB, consulta. Endpoint Amazon DocumentDB



Utilizzando l'endpoint del cluster, puoi connetterti al cluster in modalità set di repliche. Puoi quindi utilizzare le funzionalità integrate del driver delle preferenze di lettura. Nell'esempio seguente, specificando `/?replicaSet=rs0` indichi all'SDK che desideri connetterti come set di repliche. Se ometti `/?replicaSet=rs0'`, il client instrada tutte le richieste all'endpoint del cluster, ovvero all'istanza primaria.

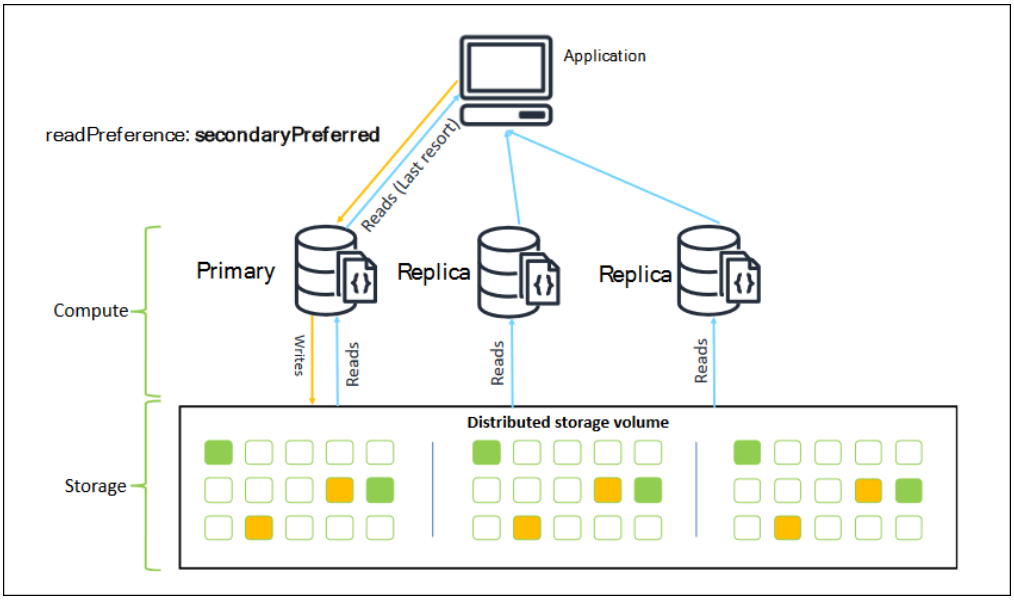
```
## Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ## replica set and specify the read preference as secondary preferred
   client = pymongo.MongoClient('mongodb://<user-name>:<password>@mycluster.node.us-
east-1.docdb.amazonaws.com:27017/?replicaSet=rs0')
```

Il vantaggio della connessione come set di repliche è che consente all'SDK di rilevare automaticamente la topografia del cluster, anche quando le istanze vengono aggiunte o rimosse dal cluster. Puoi quindi utilizzare il cluster in modo più efficiente instradando le richieste di lettura alle istanze di replica.

Quando ti connetti come set di repliche, puoi specificare `readPreference` per la connessione. Se specifichi una preferenza di lettura di `secondaryPreferred`, il client instrada le query di lettura alle repliche e le query di scrittura all'istanza primaria (come nel diagramma seguente). Le risorse del cluster vengono in tal modo utilizzate meglio. Per ulteriori informazioni, consulta Leggi le opzioni di preferenza.

```
## 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')
```



Le letture dalle repliche di Amazon DocumentDB sono alla fine coerenti. Restituiscono i dati nello stesso ordine in cui sono stati scritti sull'istanza primaria e spesso con un ritardo di replica inferiore a 50 ms. Puoi monitorare il ritardo di replica del tuo cluster utilizzando le metriche Amazon CloudWatch e. `DBInstanceReplicaLag` `DBClusterReplicaLagMaximum` Per ulteriori informazioni, consulta Monitoraggio di Amazon DocumentDB con CloudWatch.

A differenza della tradizionale architettura di database monolitica, Amazon DocumentDB separa storage ed elaborazione. Data questa architettura moderna, ti consigliamo di dimensionare la lettura sulle istanze di replica. Le letture sulle istanze di replica non bloccano le scritture replicate dall'istanza primaria. Puoi aggiungere fino a 15 istanze di replica di lettura in un cluster e dimensionare fino a milioni di letture al secondo.

Il vantaggio principale della connessione come set di repliche e della distribuzione delle letture alle repliche è quello di aumentare il numero delle risorse complessive nel cluster disponibili per l'applicazione. Come best practice, connettiti come set di repliche. Ciò è particolarmente utile nei seguenti scenari:

- Stai utilizzando quasi il 100% di CPU sull'istanza primaria.
- Il numero di riscontri nella cache del buffer è vicino a zero.
- Raggiungi i limiti di connessione o cursore per una singola istanza.

L'aumento delle dimensioni di un'istanza cluster è un'opzione e, in alcuni casi, può essere il modo migliore per dimensionare il cluster. Tuttavia, devi anche considerare come utilizzare al meglio le repliche già presenti nel cluster. In questo modo puoi incrementare le dimensioni senza aumentare i costi derivanti dall'utilizzo di un tipo di istanza più grande. Ti consigliamo inoltre di monitorare e segnalare questi limiti (ovvero `CPUUtilizationDatabaseConnections`, `eBufferCacheHitRatio`) utilizzando CloudWatch allarmi in modo da sapere quando una risorsa viene utilizzata pesantemente.

Per ulteriori informazioni, consulta i seguenti argomenti:

- Le migliori pratiche per Amazon DocumentDB
- Quote Amazon DocumentDB

Utilizzo di connessioni cluster

Considera lo scenario in cui si utilizzano tutte le connessioni del cluster. Ad esempio, un'istanza `r5.2xlarge` ha un limite di 4.500 connessioni (e 450 cursori aperti). Se crei un cluster Amazon DocumentDB a tre istanze e ti connetti solo all'istanza primaria utilizzando l'endpoint del cluster, i limiti del cluster per le connessioni aperte e i cursori sono rispettivamente di 4.500 e 450. Puoi raggiungere questi limiti se stai creando applicazioni che utilizzano molti processi di lavoro che vengono attivati in container. I container aprono una serie di connessioni tutte in una volta sola e saturano il cluster.

Invece, puoi connetterti al cluster Amazon DocumentDB come set di repliche e distribuire le tue letture alle istanze di replica. Puoi quindi triplicare in modo efficace il numero di connessioni e cursori disponibili nel cluster a 13.500 e 1.350 rispettivamente. L'aggiunta di più istanze al cluster aumenta solo il numero di connessioni e cursori per carichi di lavoro di lettura. Se devi aumentare il numero di connessioni per le scritture nel cluster, ti consigliamo di aumentare le dimensioni dell'istanza.

> **Note**
>
> Il numero di connessioni le istanze `large`, `xlarge` e `2xlarge` aumenta con le dimensioni dell'istanza fino a 4.500. Il numero massimo di connessioni per istanza per le istanze `4xlarge` o superiori è 4.500. Per ulteriori informazioni sui limiti in base ai tipi di istanza, consulta Quote delle istanze.

In genere non è consigliabile connettersi al cluster utilizzando la preferenza di lettura `secondary` perché se non ci sono istanze di replica nel cluster, le letture non vanno a buon fine. Ad esempio, supponiamo di avere un cluster Amazon DocumentDB a due istanze con una primaria e una replica. Se la replica presenta un problema, le richieste di lettura da un pool di connessioni impostato come `secondary` non vanno a buon fine. Il vantaggio di `secondaryPreferred` è che se il client non è in grado di trovare un'istanza di replica idonea a cui connettersi, esegue il fallback sull'istanza primaria per le letture.

Pool di connessioni multipli

In alcuni scenari, le letture in un'applicazione devono avere una coerenza di lettura dopo la scrittura, che può essere eseguita solo dall'istanza principale in Amazon DocumentDB. In questi scenari, è possibile creare due pool di connessioni client: uno per le scritture e uno per le letture che richiedono la consistenza lettura dopo scrittura. Per effettuare questa operazione, il codice sarà simile al seguente.

```
## 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')
```

Un'altra opzione consiste nel creare un singolo pool di connessioni e sovrascrivere la preferenza di lettura per una determinata raccolta.

```
##Specify the collection and set the read preference level for that collection
col = db.review.with_options(read_preference=ReadPreference.SECONDARY_PREFERRED)
```

Riepilogo

Per utilizzare al meglio le risorse del cluster, connettiti utilizzando la modalità replica set. Se è adatto per la tua applicazione, puoi dimensionare la lettura dell'applicazione distribuendo le letture alle istanze di replica.

Connessione a un cluster Amazon DocumentDB da Studio 3T

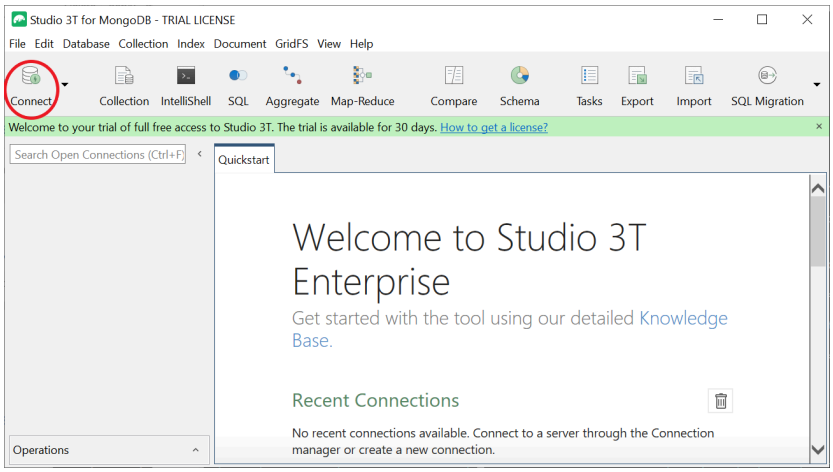
Studio 3T è una GUI e IDE popolare per sviluppatori e ingegneri dei dati che lavorano con MongoDB. Offre diverse potenti funzionalità, visualizzazioni ad albero, tabella e JSON dei dati, facilità import/export in CSV, JSON, SQL e un'opzione di interrogazione flessibile BSON/mongodump, un'interfaccia utente visiva drag-and-drop, una shell mongo integrata con completamento automatico, un editor di pipeline di aggregazione e supporto per query SQL.

Prerequisiti

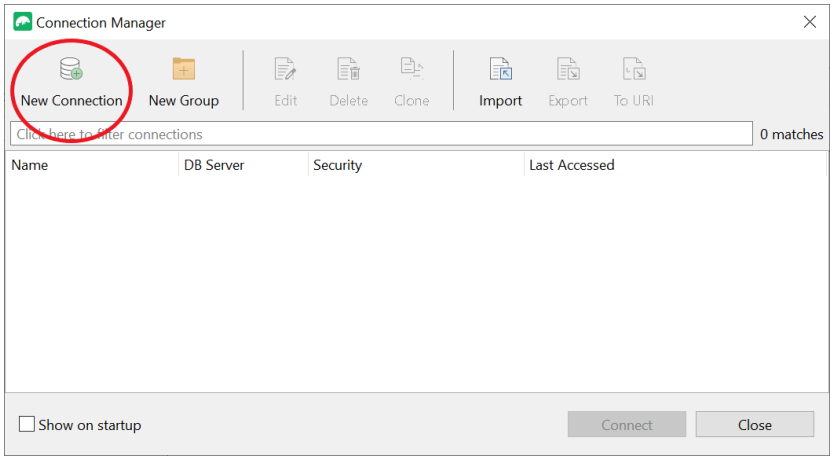
- Se non disponi già di un cluster Amazon DocumentDB che utilizza Amazon EC2 come host, bastion/jump consulta. Connettiti tramite Amazon EC2
- Se non disponi di Studio 3T, scaricalo e installalo.

Connect con Studio 3T

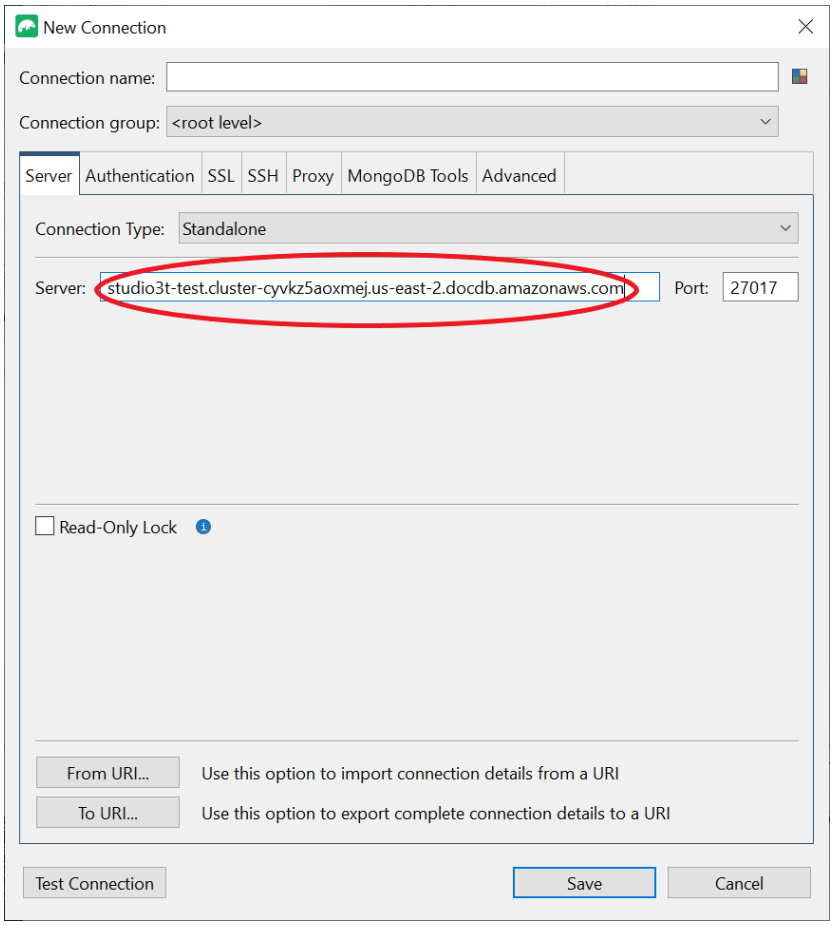
1. Scegli Connect nell'angolo in alto a sinistra della barra degli strumenti.



2. Scegli Nuova connessione nell'angolo in alto a sinistra della barra degli strumenti.

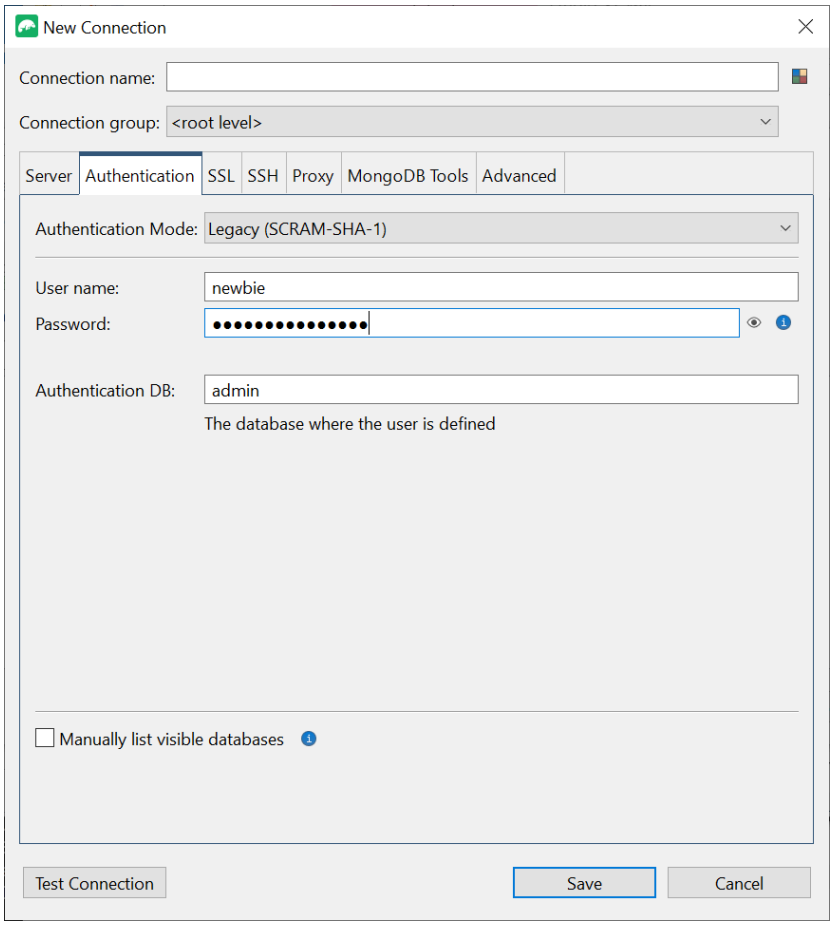


3. Nella scheda Server, nel campo Server, inserisci le informazioni sull'endpoint del cluster.

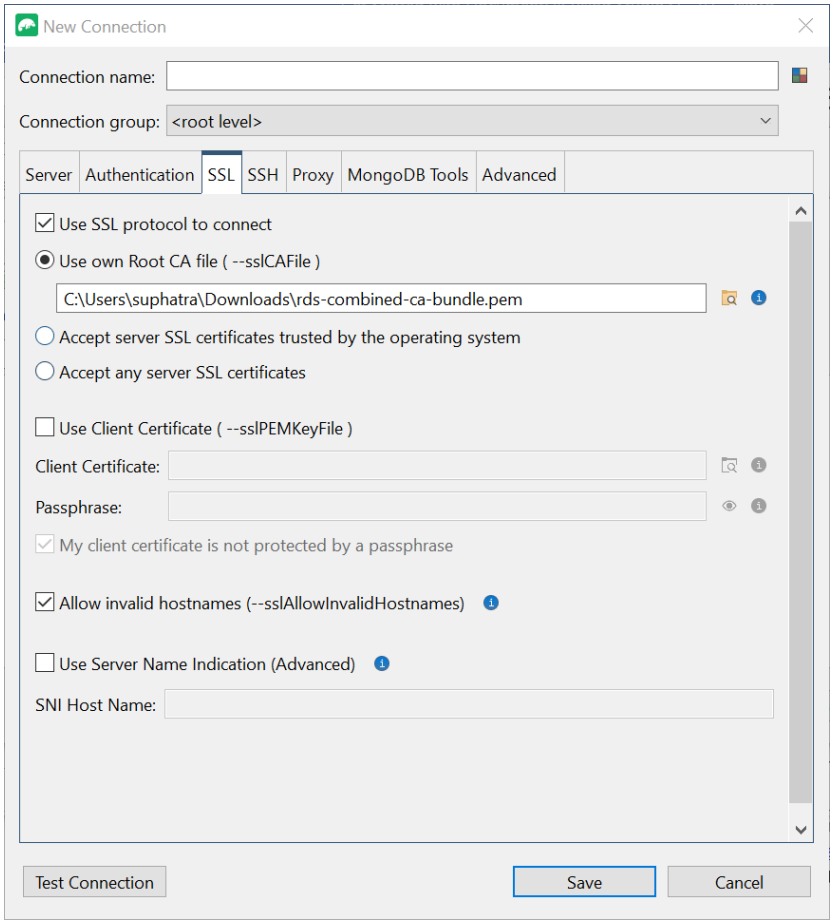


> **Note**
>
> Per trovare l'endpoint del cluster, consultaIndividuazione dell'endpoint di un'istanza.

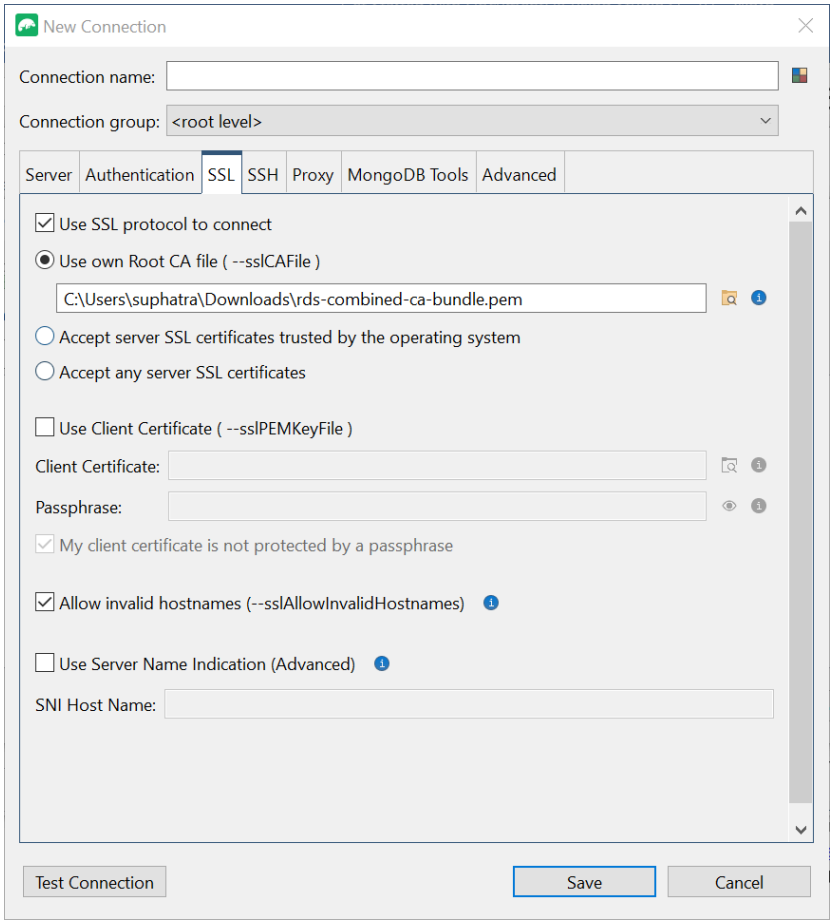
4. Scegli la scheda Autenticazione e seleziona Legacy nel menu a discesa per la modalità di autenticazione.



5. Inserisci il tuo nome utente e le tue credenziali nei campi Nome utente e Password.
6. Scegli la scheda SSL e seleziona la casella Usa il protocollo SSL per connetterti.



7. Scegli Usa il tuo file Root CA. Aggiungi quindi il certificato Amazon DocumentDB (puoi saltare questo passaggio se SSL è disabilitato sul tuo cluster DocumentDB). Seleziona la casella per consentire nomi host non validi.



> **Note**
>
> Non hai il certificato? Puoi scaricarlo con il seguente comando:
> `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

8. Se ti connetti da una macchina client esterna ad Amazon VPC, devi creare un tunnel SSH. Lo farai nella scheda SSH.

   a. Seleziona la casella Usa tunnel SSH e inserisci l'indirizzo SSH nel campo Indirizzo SSH. Questa è la tua istanza Public DNS (IPV4). Puoi ottenere questo URL dalla tua console di gestione Amazon EC2.

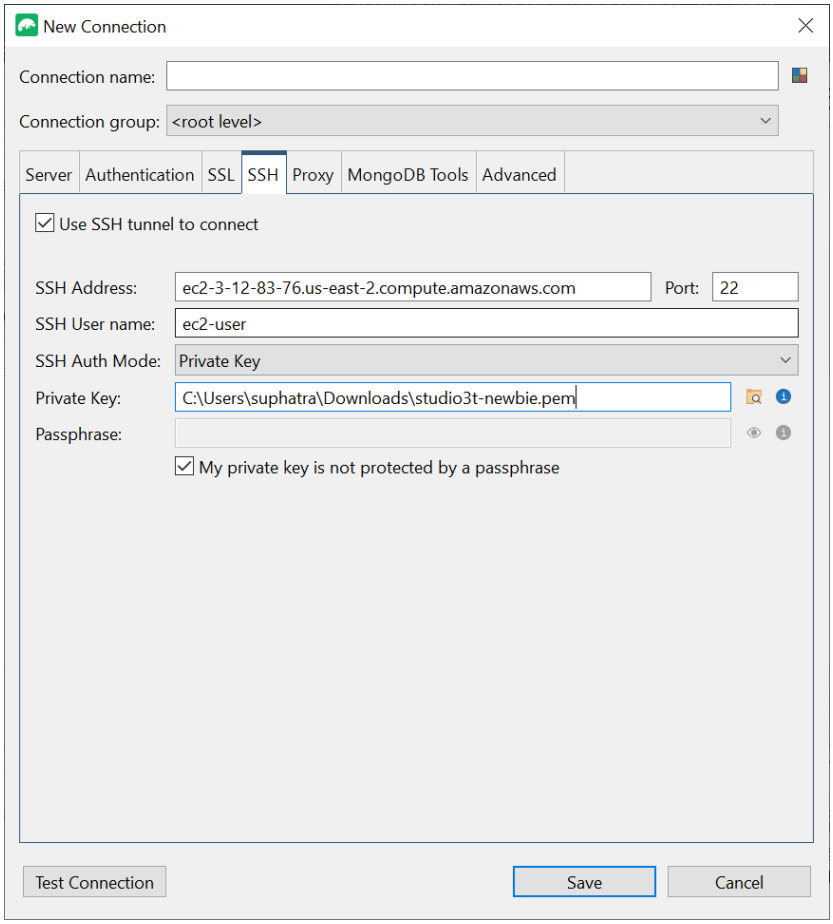
   b. Inserisci il tuo nome utente. Questo è il nome utente della tua istanza Amazon EC2

   c. Per la modalità di autenticazione SSH, seleziona Chiave privata. Nel campo Chiave privata, scegli l'icona di ricerca dei file da individuare e scegli la chiave privata della tua istanza

Amazon EC2. Questo è il file.pem (key pair) che hai salvato durante la creazione dell'istanza nella console Amazon EC2.

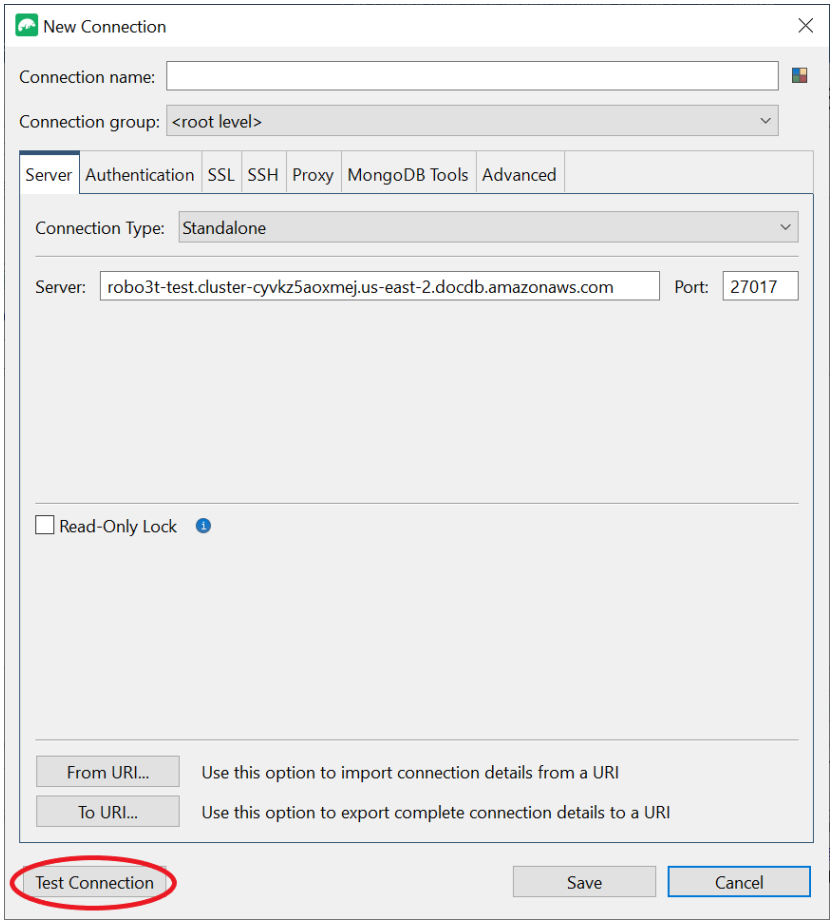
d. Se utilizzi un computer Linux/macOS client, potresti dover modificare le autorizzazioni della tua chiave privata utilizzando il seguente comando:

```
chmod 400 /fullPathToYourPemFile/<yourKey>.pem
```

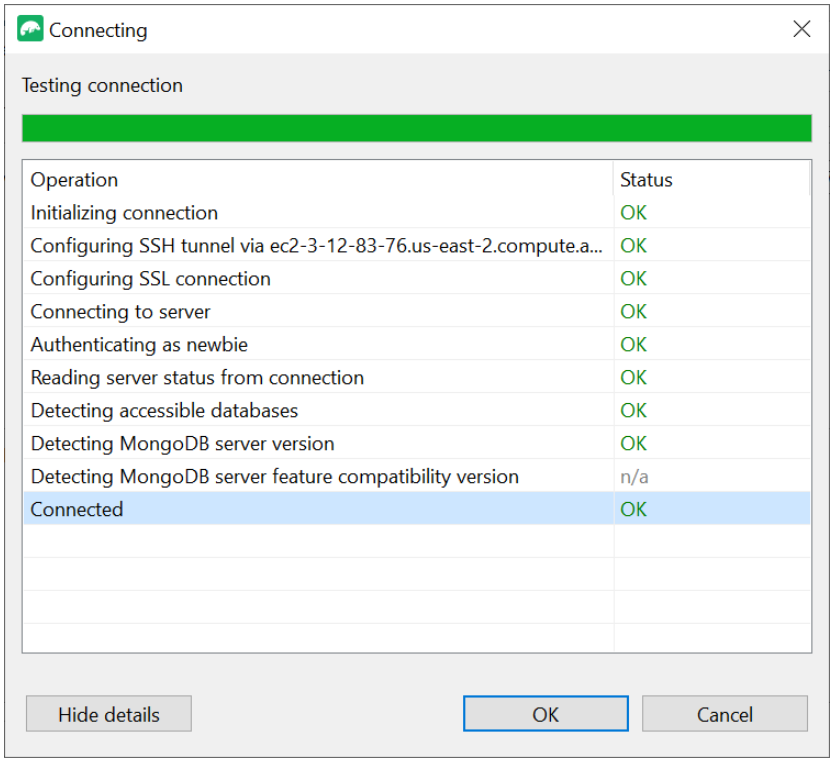


> **Note**
>
> Questa istanza Amazon EC2 deve trovarsi nello stesso gruppo di sicurezza e VPC di Amazon del cluster DocumentDB. Puoi ottenere l'indirizzo SSH, il nome utente e la chiave privata dalla tua console di gestione Amazon EC2.

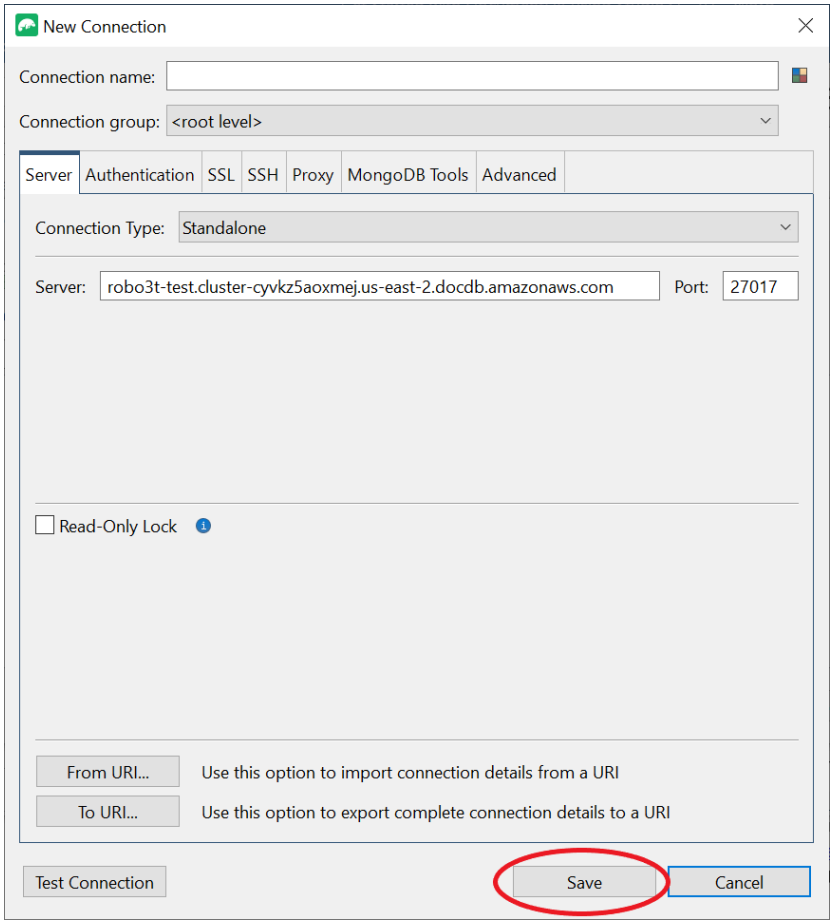
9. Ora verifica la tua configurazione scegliendo il pulsante Test di connessione.



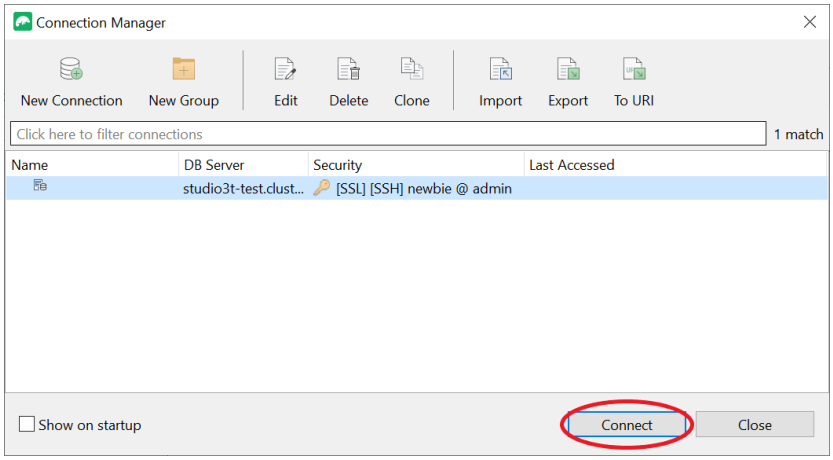
10. Una finestra di diagnostica dovrebbe caricare una barra verde per indicare che il test ha avuto esito positivo. Ora scegli OK per chiudere la finestra di diagnostica.



11. Scegli Salva per salvare la connessione per utilizzi futuri.



12. Ora seleziona il tuo cluster e scegli Connect.



Complimenti! Ora sei connesso correttamente al tuo cluster Amazon DocumentDB tramite Studio 3T.

Connect ad Amazon DocumentDB tramite DataGrip

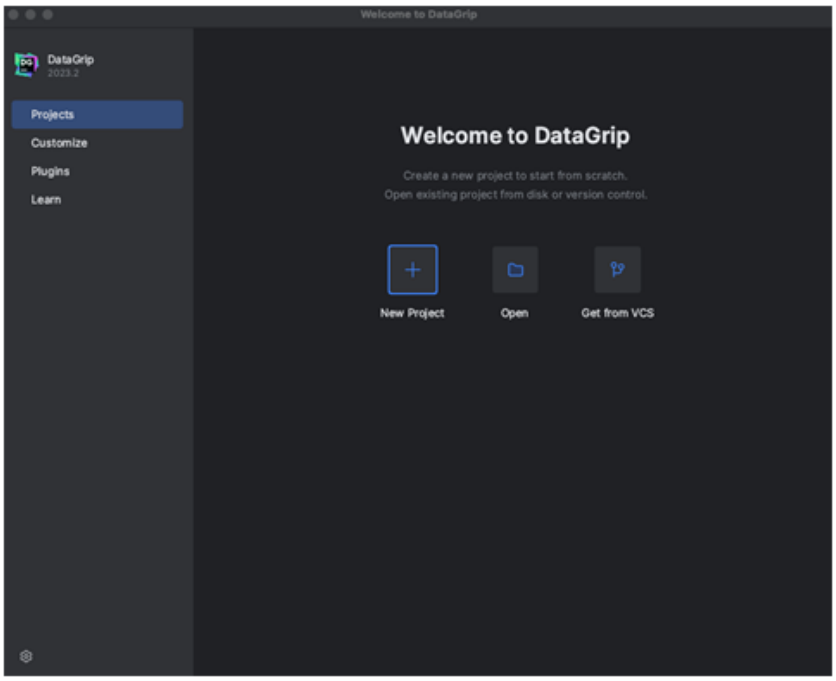
DataGripè un potente ambiente di sviluppo integrato (IDE) che supporta vari sistemi di database, incluso Amazon DocumentDB. Questa sezione illustra i passaggi per connettersi al cluster Amazon DocumentDB utilizzando DataGrip, in modo da gestire e interrogare facilmente i dati utilizzando un'interfaccia grafica.

Prerequisiti

- DataGrip IDE installato sulla tua macchina. Puoi scaricarlo da JetBrains.
- Un'istanza Amazon EC2 in esecuzione nello stesso VPC del cluster Amazon DocumentDB. Utilizzerai questa istanza per stabilire un tunnel sicuro dal tuo computer locale ad Amazon DocumentDBCluster. Segui le istruzioni su come farlo. Connettiti tramite Amazon EC2
- Alternativa a un'istanza Amazon EC2, a una connessione VPN o se stai già accedendo alla tua AWS infrastruttura utilizzando una VPN sicura. Se preferisci questa opzione, segui le istruzioni per accedere in modo sicuro ad Amazon AWS Client VPN DocumentDB utilizzando.

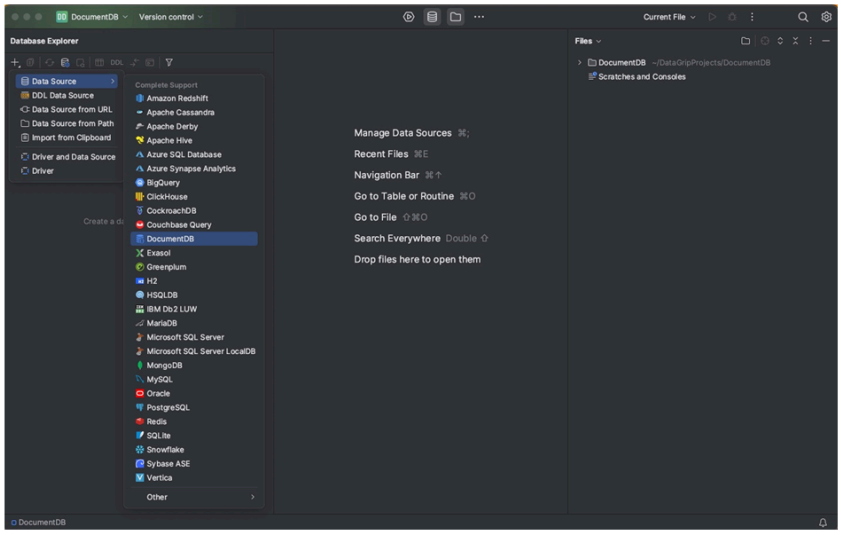
Connect usando DataGrip

1. Avvia DataGrip sul tuo computer e crea un nuovo progetto.

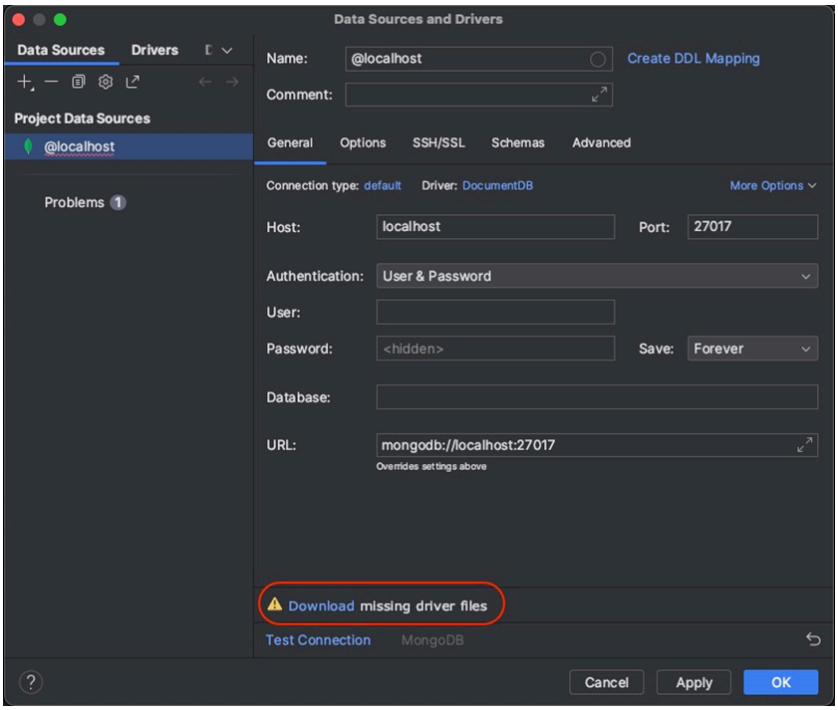


2. Aggiungi una nuova fonte di dati utilizzando uno dei seguenti modi:
   a. Dal menu principale, accedi a File — Nuovo — Origine dati e seleziona DocumentDB

b. In Database Explorer, scegli la nuova icona (+) nella barra degli strumenti. Vai a Data Source e seleziona DocumentDB.



3. Nella pagina Fonti dati della scheda Generale, controlla se c'è il link Scarica i file dei driver mancanti nella parte inferiore dell'area delle impostazioni di connessione. Scegli questo link per scaricare i driver necessari per interagire con un database. Per un link per il download diretto, consulta i driver JetBrains JDBC.



4. Nella scheda Generale, specifica i dettagli della connessione:

a. Nel campo Host, specifica l'endpoint del cluster Amazon DocumentDB.

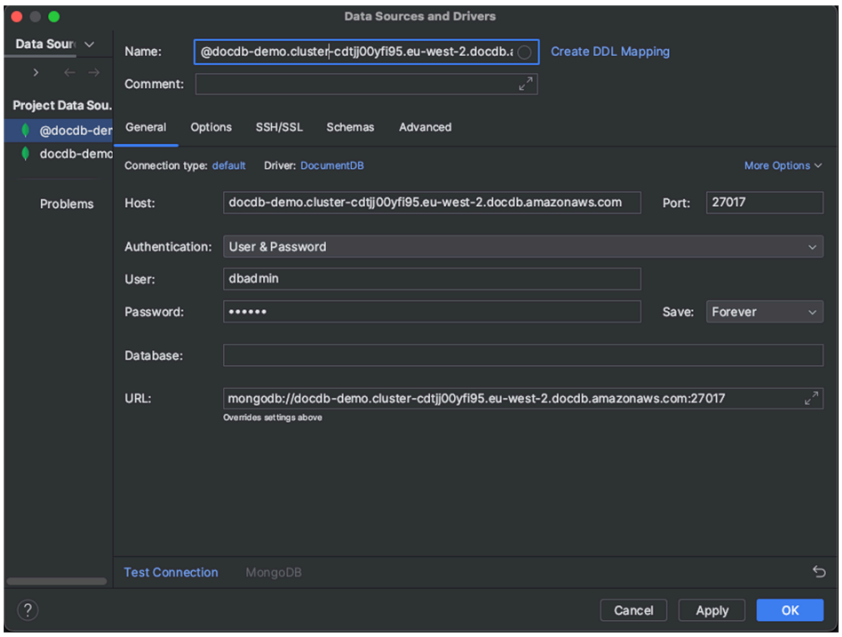
b. La porta è già impostata su 27017. Modificala se il cluster è stato distribuito su una porta diversa.

c. Per l'autenticazione, scegli Utente e password.

d. Inserisci le informazioni relative al nome utente e alla password.

e. Il campo Database è facoltativo. È possibile specificare il database a cui connettersi.

f. Il campo URL viene completato automaticamente man mano che aggiungi i dettagli precedenti.



5. Nella SSH/SSLscheda, abilita Usa tunnel SSH, quindi scegli l'icona per aprire la finestra di dialogo di configurazione SSH. Immetti le seguenti informazioni:

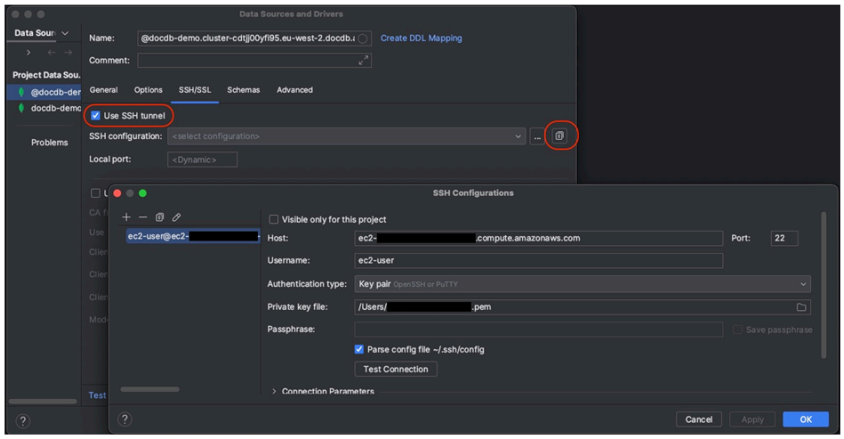
a. nel campo Host, inserisci il nome host della tua istanza Amazon EC2.

b. Inserisci il nome utente e la password per la tua istanza Amazon EC2.

c. Per Tipo di autenticazione, scegli Coppia di chiavi.

d. Inserisci il tuo file di chiave privata.

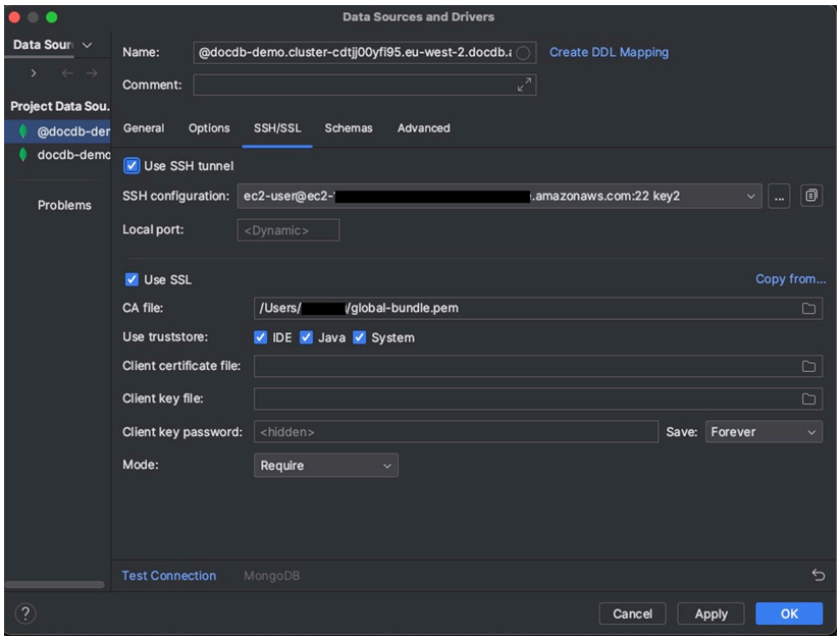
> **Note**
>
> Se utilizzi l'opzione VPN, non è necessario configurare il tunnel SSH.



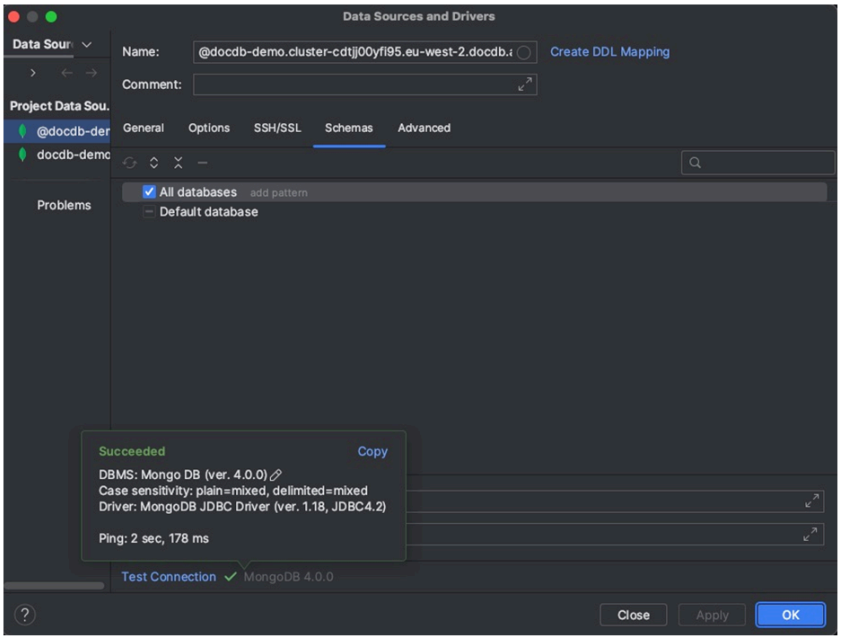
6. Nella SSH/SSLscheda, abilita Usa SSL. Nel campo File CA, inserisci la posizione del `global-bundle.pem` file sul tuo computer. Per Modalità, lascia l'opzione Richiedi.

> **Note**
>
> Puoi scaricare il certificato da questa posizione o con questo comando: wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem

> **Note**
>
> Se ti connetti al cluster elastico di Amazon DocumentDB, non devi specificare il file CA. Lascia selezionata l'opzione Usa SSL e tutte le altre opzioni ai valori predefiniti.



7. Nella scheda Schemi, scegli Tutti i database o inserisci il filtro «*: *» nel campo Schema. Scegli Test Connection per testare la connessione.



8. Una volta che la connessione è stata testata correttamente, scegli OK per salvare la configurazione dell'origine dati.

DataGrip features

DataGrip offre diverse funzionalità per aiutarti a lavorare con Amazon DocumentDB in modo efficiente:

- SQL Editor: scrivi ed esegui SQL-like query sulle tue raccolte DocumentDB utilizzando l'editor SQL in. DataGrip
- Visual Query Builder: utilizza il generatore di query visive per creare query graficamente senza scrivere codice SQL.
- Gestione dello schema: gestisci facilmente lo schema del tuo database, inclusa la creazione, la modifica e l'eliminazione delle raccolte.
- Visualizzazione dei dati: visualizza e analizza i dati utilizzando vari strumenti di visualizzazione disponibili in. DataGrip
- Esportazione e importazione di dati: trasferisci dati tra Amazon DocumentDB e altri database utilizzando le funzionalità DataGrip di esportazione e importazione di.

Consulta la DataGrip documentazione ufficiale per funzionalità e suggerimenti più avanzati su come lavorare con Amazon DocumentDB e altri sistemi di database.

Connettiti tramite Amazon EC2

Questa sezione descrive come configurare la connettività tra un cluster Amazon DocumentDB e Amazon EC2 e accedere al cluster Amazon DocumentDB dall'istanza Amazon EC2.

Esistono due opzioni per configurare la connessione EC2:

- Connetti automaticamente la tua istanza EC2 a un database Amazon DocumentDB: utilizza la funzionalità di connessione automatica nella console EC2 per configurare automaticamente la connessione tra l'istanza EC2 e un database Amazon DocumentDB nuovo o esistente. Questa connessione consente al traffico di viaggiare tra l'istanza EC2 e il database Amazon DocumentDB. Questa opzione viene in genere utilizzata per testare e creare nuovi gruppi di sicurezza.
- Connetti manualmente l'istanza EC2 al database Amazon DocumentDB: configura la connessione tra l'istanza EC2 al database Amazon DocumentDB configurando e assegnando manualmente i gruppi di sicurezza per riprodurre la configurazione creata dalla funzionalità di connessione automatica. Questa opzione viene in genere utilizzata per modificare impostazioni più avanzate e utilizzare i gruppi di sicurezza esistenti.

Prerequisiti

Indipendentemente dall'opzione, e prima di creare il tuo primo cluster Amazon DocumentDB, devi fare quanto segue:

Crea un account Amazon Web Services (AWS)

Prima di poter iniziare a utilizzare Amazon DocumentDB, devi disporre di un account Amazon Web Services (AWS). L' AWS account è gratuito. Paghi solo per i servizi e le risorse che utilizzi.

Se non ne hai uno Account AWS, completa i seguenti passaggi per crearne uno.

Per iscriverti a Account AWS

1. Aperta https://signin.aws.amazon.com/signup? request_type=registrati.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata o un messaggio di testo e ti verrà chiesto di inserire un codice di verifica attraverso la tastiera del telefono.

   Quando ti iscrivi a un Account AWS, viene creato un. Utente root dell'account AWS L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

Facoltativamente, imposta le autorizzazioni AWS Identity and Access Management (IAM) necessarie.

L'accesso per gestire le risorse di Amazon DocumentDB come cluster, istanze e gruppi di parametri del cluster richiede credenziali che AWS possono essere utilizzate per autenticare le tue richieste. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per Amazon DocumentDB.

1. Nella barra di ricerca di Console di gestione AWS, inserisci **`IAM`** e seleziona IAM nel menu a discesa che appare.
2. Una volta entrati nella console IAM, seleziona Utenti dal pannello di navigazione.
3. Seleziona il tuo nome utente.
4. Scegli Add Permissions (Aggiungi autorizzazioni).
5. Seleziona Attach existing policies directly (Collega direttamente le policy esistenti).

6. `AmazonDocDBFullAccess`Digita la barra di ricerca e selezionala una volta visualizzata nei risultati della ricerca.
7. Scegli Prossimo: Rivedi.
8. Scegli Add Permissions (Aggiungi autorizzazioni).

Crea un Amazon Virtual Private Cloud (Amazon VPC)

A seconda del luogo in cui Regione AWS ti trovi, potresti avere già creato o meno un VPC predefinito. Se non disponi di un VPC predefinito, completa il passaggio 1 della Guida introduttiva ad Amazon VPC nella Amazon VPC User Guide. Questa operazione richiederà meno di cinque minuti.

Connetti Amazon EC2 automaticamente

Argomenti

- Connetti automaticamente un'istanza EC2 a un nuovo database Amazon DocumentDB
- Connetti automaticamente un'istanza EC2 a un database Amazon DocumentDB esistente
- Panoramica della connettività automatica con un'istanza EC2
- Visualizzazione delle risorse di calcolo connesse

Prima di configurare una connessione tra un'istanza EC2 e un nuovo database Amazon DocumentDB, assicurati di soddisfare i requisiti descritti in. Panoramica della connettività automatica con un'istanza EC2 Se apporti modifiche ai gruppi di sicurezza dopo aver configurato la connettività, le modifiche potrebbero influire sulla connessione tra l'istanza EC2 e il database Amazon DocumentDB.

> **Note**
>
> Puoi configurare automaticamente una connessione tra un'istanza EC2 e un database Amazon DocumentDB solo utilizzando. Console di gestione AWS Non puoi configurare una connessione automaticamente con l'API AWS CLI o Amazon DocumentDB.

Connetti automaticamente un'istanza EC2 a un nuovo database Amazon DocumentDB

La procedura seguente presuppone che tu abbia completato i passaggi indicati nell'Prerequisitiargomento.

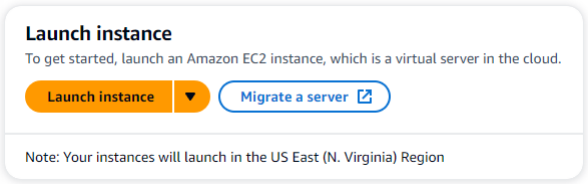
Fasi

- Passaggio 1: creare un'istanza Amazon EC2
- Passaggio 2: creare un cluster Amazon DocumentDB
- Passaggio 3: connettiti alla tua istanza Amazon EC2
- Passaggio 4: installa MongoDB Shell
- Passaggio 5: gestione di Amazon DocumentDB TLS
- Passaggio 6: connettiti al tuo cluster Amazon DocumentDB
- Fase 7: Inserimento e interrogazione dei dati
- Fase 8: Esplora

Passaggio 1: creare un'istanza Amazon EC2

In questo passaggio, creerai un'istanza Amazon EC2 nella stessa regione e Amazon VPC che utilizzerai successivamente per il provisioning del tuo cluster Amazon DocumentDB.

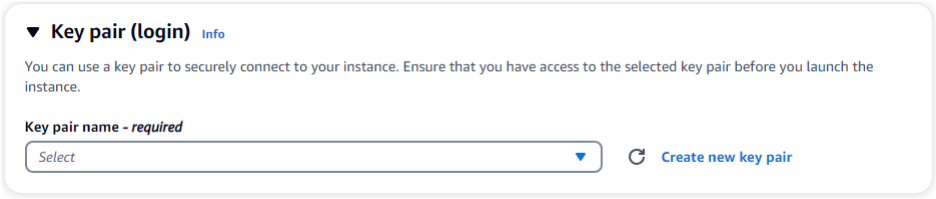
1. Sulla console Amazon EC2, scegli Launch instance.



2. Inserisci un nome o un identificatore nel campo Nome situato nella sezione Nome e tag.
3. Nell'elenco a discesa Amazon Machine Image (AMI), individua Amazon Linux 2 AMI e sceglila.

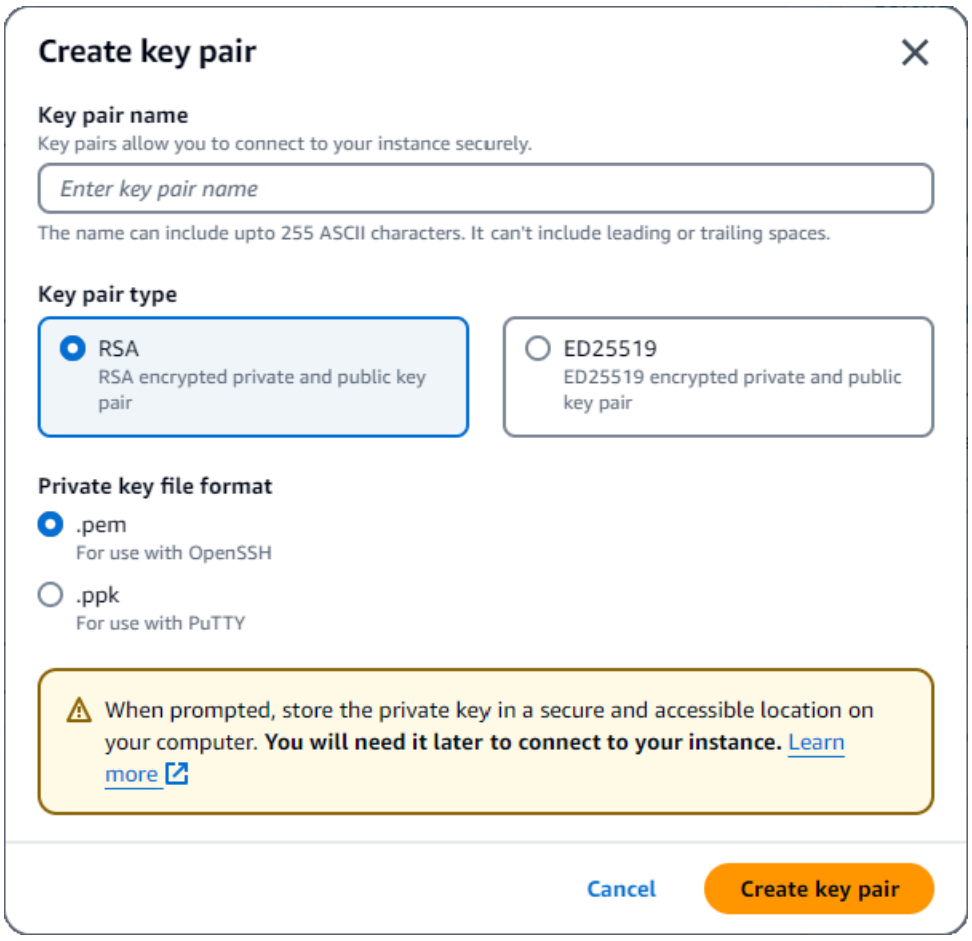


4. Individua e scegli t3.micro nell'elenco a discesa Tipo di istanza.
5. Nella sezione Coppia di chiavi (accesso), inserisci l'identificativo di una coppia di chiavi esistente o scegli Crea nuova coppia di chiavi.



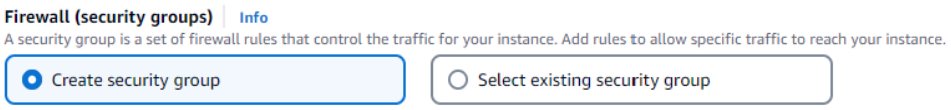
È necessario fornire una coppia di chiavi Amazon EC2.

- Se disponi di una coppia di chiavi Amazon EC2:
  1. Seleziona una coppia di chiavi, scegline una dall'elenco.
  2. Devi avere già il file della chiave privata (.pem o .ppk) disponibile per accedere alla tua istanza Amazon EC2.
- Se non disponi di una coppia di chiavi Amazon EC2:
  1. Scegli Crea nuova coppia di chiavi, viene visualizzata la finestra di dialogo Crea coppia di chiavi.
  2. Inserite un nome nel campo Nome coppia di chiavi.
  3. Scegliete il tipo di coppia di chiavi e il formato del file della chiave privata.
  4. Scegliere Create key pair (Crea coppia di chiavi).



> **Note**
>
> Per motivi di sicurezza, consigliamo vivamente di utilizzare una coppia di chiavi per la connettività SSH e Internet all'istanza EC2.

6. Facoltativo: nella sezione Impostazioni di rete, in Firewall (gruppi di sicurezza), scegli Crea gruppo di sicurezza.



Scegli Crea gruppo di sicurezza (controlla tutte le regole di autorizzazione del traffico che si applicano alla tua connettività EC2).

> **Note**
>
> Se desideri utilizzare un gruppo di sicurezza esistente, segui le istruzioni in Connetti Amazon EC2 manualmente.

7. Nella sezione Riepilogo, rivedi la tua configurazione EC2 e scegli l'istanza Launch, se corretta.

Passaggio 2: creare un cluster Amazon DocumentDB

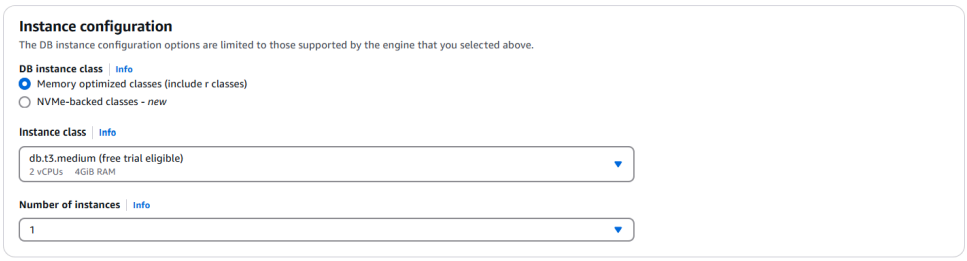
Durante il provisioning dell'istanza Amazon EC2, crea il tuo cluster Amazon DocumentDB.

1. Accedi alla console Amazon DocumentDB e scegli Clusters dal pannello di navigazione.
2. Scegli Crea.
3. Lascia l'impostazione del tipo di cluster come impostazione predefinita, Instance Based Cluster.
4. Nella configurazione del cluster, per l'identificatore del cluster, inserisci un nome univoco. Tieni presente che la console cambierà tutti i nomi dei cluster in lettere minuscole indipendentemente da come vengono immessi.

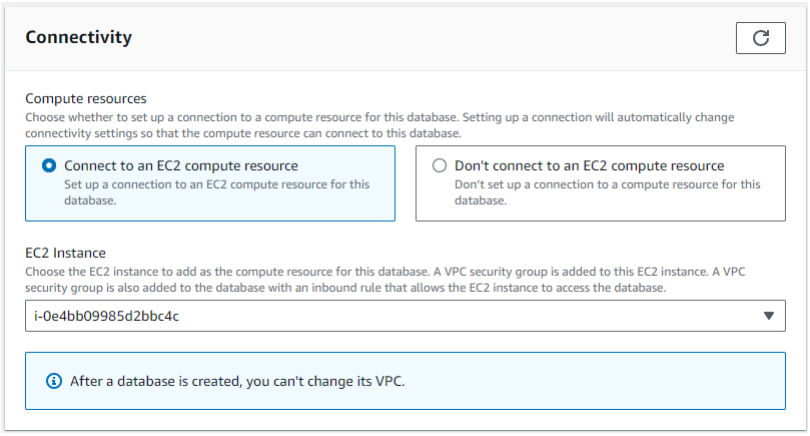
   Lascia la versione Engine al valore predefinito di 5.0.0.
5. Per la configurazione dello storage in cluster, lascia l'impostazione predefinita di Amazon DocumentDB Standard.
6. Nella configurazione dell'istanza:

   - Per la classe di istanza DB, scegli le classi ottimizzate per la memoria (includi le classi r) (questa è l'impostazione predefinita).

     L'altra opzione di istanza sono NVMe-backed le classi. Per ulteriori informazioni, consulta NVMe-backed istanze.
   - Per la classe Instance, scegli il tipo di istanza più adatto alle tue esigenze. Per una spiegazione più dettagliata delle classi di istanze, consultaSpecifiche della classe di istanza.
   - Per il numero di istanze, scegliete un numero che rispecchi al meglio le vostre esigenze. Ricorda che più basso è il numero, minore è il costo e minore è il read/write volume che può essere gestito dal cluster.



7. Per Connettività, scegli Connetti a una risorsa di calcolo EC2. Questa è l'istanza EC2 creata nel passaggio 1.



> **Note**
>
> La connessione a una risorsa di calcolo EC2 crea automaticamente un gruppo di sicurezza per la connessione della risorsa di calcolo EC2 al cluster Amazon DocumentDB. Quando hai completato la creazione del cluster e desideri visualizzare il gruppo di sicurezza appena creato, vai all'elenco dei cluster e scegli l'identificatore del cluster. Nella scheda Connettività e sicurezza, vai su Gruppi di sicurezza e trova il tuo gruppo in Nome del gruppo di sicurezza (ID). Conterrà il nome del tuo cluster e avrà un aspetto simile a questo:`docdb-ec2-docdb-2023-12-11-21-33-41:i-0e4bb09985d2bbc4c (sg-0238e0b0bf0f73877)`.

8. Nella sezione Autenticazione, inserisci un nome utente per l'utente principale, quindi scegli Gestito autonomamente. Inserisci una password, quindi confermala.

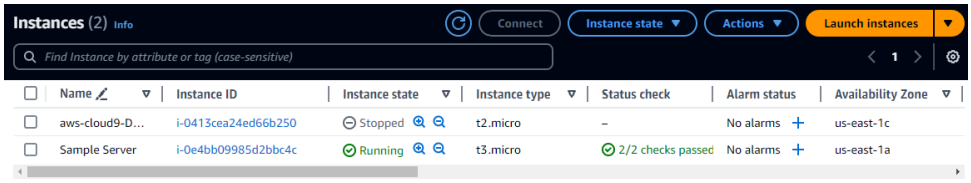
   Se invece hai scelto Managed in AWS Secrets Manager, consulta Gestione delle password con Amazon DocumentDB e AWS Secrets Manager per ulteriori informazioni.

9. Scegli Crea cluster.

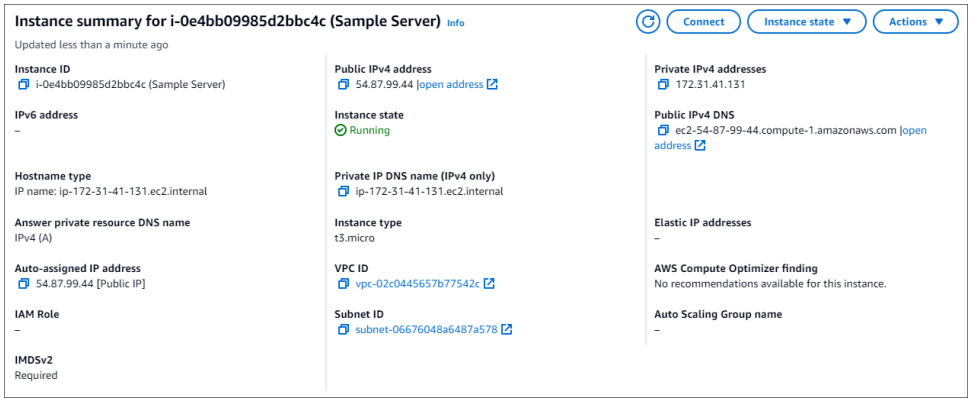
Passaggio 3: connettiti alla tua istanza Amazon EC2

Per installare mongo shell, devi prima connetterti alla tua istanza Amazon EC2. L'installazione di mongo shell ti consente di connetterti e di interrogare il tuo cluster Amazon DocumentDB. Completa questa procedura:

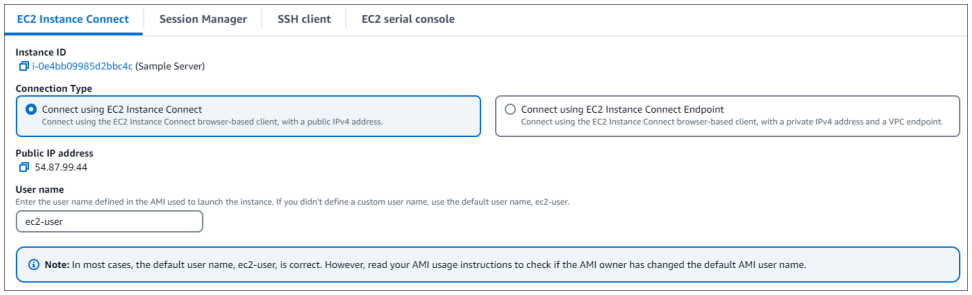
1. Sulla console Amazon EC2, accedi alle tue istanze e verifica se l'istanza appena creata è in esecuzione. In tal caso, seleziona l'istanza facendo clic sull'ID dell'istanza.



2. Scegli Connetti.



3. Sono disponibili quattro opzioni a schede per il metodo di connessione: Amazon EC2 Instance Connect, Session Manager, client SSH o console seriale EC2. Devi sceglierne una e seguirne le istruzioni. Al termine, scegli Connetti.



> **Note**
>
> Se il tuo indirizzo IP è cambiato dopo aver iniziato questa procedura dettagliata o tornerai al tuo ambiente in un secondo momento, devi aggiornare la regola in entrata del tuo gruppo di demoEC2 sicurezza per abilitare il traffico in entrata dal tuo nuovo indirizzo API.

Passaggio 4: installa MongoDB Shell

Ora puoi installare la shell MongoDB, un'utilità a riga di comando che usi per connetterti e interrogare il tuo cluster Amazon DocumentDB. Attualmente esistono due versioni di MongoDB shell: la versione più recente, mongosh, e la versione precedente, mongo shell.

> **Important**
>
> Esiste una limitazione nota con Node.js i driver precedenti alla versione 6.13.1, che attualmente non sono supportati dall'autenticazione dell'identità IAM per Amazon DocumentDB. Node.js i driver e gli strumenti che utilizzano Node.js driver (ad esempio, mongosh) devono essere aggiornati per utilizzare la versione 6.13.1 o successiva. Node.js

Segui le istruzioni seguenti per installare la shell MongoDB per il tuo sistema operativo.

On Amazon Linux

Per installare la shell MongoDB su Amazon Linux

Se non utilizzi l'autenticazione IAM e desideri utilizzare la shell MongoDB più recente (mongosh) per connetterti al tuo cluster Amazon DocumentDB, segui questi passaggi:

1. Crea il file del repository. Nella riga di comando dell'istanza EC2 che hai creato, esegui il seguente comando:

   ```
   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. Al termine, installa mongosh con una delle due seguenti opzioni di comando al prompt dei comandi:

Opzione 1: se hai scelto Amazon Linux 2023 predefinito durante la configurazione di Amazon EC2, inserisci questo comando:

```
sudo yum install -y mongodb-mongosh-shared-openssl3
```

Opzione 2: se hai scelto Amazon Linux 2 durante la configurazione di Amazon EC2, inserisci questo comando:

```
sudo yum install -y mongodb-mongosh
```

Se utilizzi l'autenticazione IAM, devi utilizzare la versione precedente della shell MongoDB (5.0) per connetterti al tuo cluster Amazon DocumentDB, segui questi passaggi:

1. Crea il file del repository. Nella riga di comando dell'istanza EC2 che hai creato, esegui il seguente comando:

```
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. Al termine, installa la shell mongodb 5.0 con la seguente opzione di comando al prompt dei comandi:

```
sudo yum install -y mongodb-org-shell
```

On Ubuntu

Per installare mongosh su Ubuntu

1. Importa la chiave pubblica che verrà utilizzata dal sistema di gestione dei pacchetti.

```
curl -fsSL https://pgp.mongodb.com/server-5.0.asc | sudo gpg --dearmor -o /usr/
share/keyrings/mongodb-server-5.0.gpg
```

2. Crea l'elenco di file `mongodb-org-5.0.list` per MongoDB utilizzando il comando appropriato per la versione di Ubuntu.

```
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. Importa e aggiorna il database locale dei pacchetti usando il seguente comando:

```
sudo apt-get update
```

4. Installa mongosh.

```
sudo apt-get install -y mongodb-mongosh
```

Per informazioni sull'installazione di versioni precedenti di MongoDB nel sistema Ubuntu, consulta Installazione di MongoDB Community Edition in Ubuntu.

On other operating systems

Per installare la shell Mongo in altri sistemi operativi, vedi l'argomento relativo all'installazione di MongoDB Community Edition nella documentazione relativa a MongoDB.

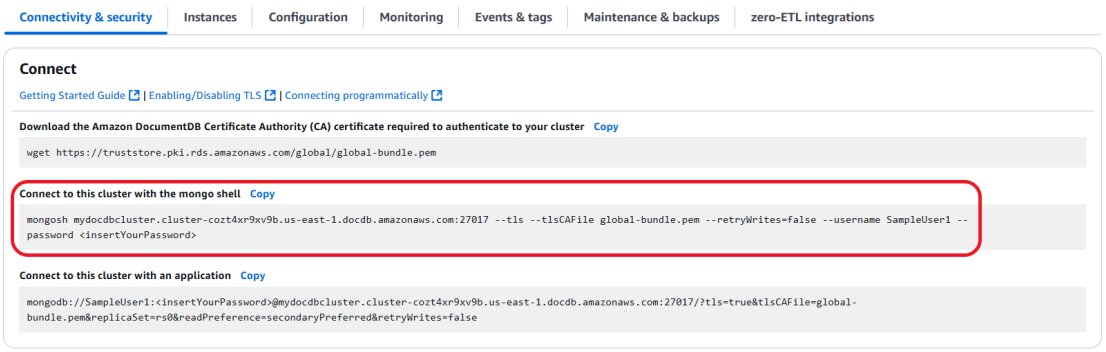
Passaggio 5: gestione di Amazon DocumentDB TLS

Scarica il certificato CA per Amazon DocumentDB con il codice seguente: `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

> **Note**
>
> Transport Layer Security (TLS) è abilitato di default per tutti i nuovi cluster Amazon DocumentDB. Per ulteriori informazioni, consulta Crittografia dei dati in transito.

Passaggio 6: connettiti al tuo cluster Amazon DocumentDB

1. Nella console Amazon DocumentDB, in Clusters, individua il cluster. Scegli il cluster che hai creato facendo clic sull'identificatore del cluster per quel cluster.
2. Nella scheda Connettività e sicurezza, individua Connetti a questo cluster con mongo shell nella casella Connect:



Copia la stringa di connessione fornita e incollala nel tuo terminale.

Apporta le seguenti modifiche:

a. Assicurati di avere il nome utente corretto nella stringa.

b. Ometti `<insertYourPassword>` in modo che ti venga richiesta la password dalla shell mongo quando ti connetti.

c. Facoltativo: se utilizzi l'autenticazione IAM o stai utilizzando la versione precedente della shell MongoDB, modifica la stringa di connessione come segue:

```
mongo --ssl --host mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1.docdb.amazonaws.com:27017 --sslCAFile global-bundle.pem --username SampleUser1 --password
```

Sostituisci `mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1` con le stesse informazioni del tuo cluster.

3. Premi invio nel tuo terminale. Ora ti verrà richiesta la password. Inserisci la password.
4. Quando inserisci la password e viene visualizzata la `rs0 [direct: primary] <env-name>>` richiesta, sei connesso correttamente al tuo cluster Amazon DocumentDB.

Per risolvere i problemi di connessione, consultaRisoluzione dei problemi di Amazon DocumentDB.

Fase 7: Inserimento e interrogazione dei dati

Ora che sei connesso al tuo cluster, puoi eseguire alcune query per acquisire familiarità con l'utilizzo di un database di documenti.

1. Per inserire un singolo documento, inserisci quanto segue:

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

Si ottiene il seguente risultato:

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

2. Puoi leggere il documento che hai scritto con il `findOne()` comando (perché restituisce solo un singolo documento). Inserisci quanto segue:

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

Si ottiene il seguente risultato:

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

3. Per eseguire qualche altra domanda, considera un caso d'uso dei profili di gioco. Innanzitutto, inserisci alcune voci in una raccolta intitolata`profiles`. Inserisci quanto segue:

```
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 }
])
```

Si ottiene il seguente risultato:

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

4. Usa il `find()` comando per restituire tutti i documenti nella raccolta dei profili. Inserisci quanto segue:

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

Otterrai un risultato che corrisponderà ai dati che hai digitato nel passaggio 3.

5. Usa una query per un singolo documento usando un filtro. Inserisci quanto segue:

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

   Si ottiene il seguente risultato:

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

6. Ora proviamo a trovare un profilo e modificarlo usando il `findAndModify` comando. Daremo all'utente Matt altri 10 punti con il seguente codice:

```
db.profiles.findAndModify({
        query: { name: "Matt", status: "active"},
        update: { $inc: { score: 10 } }
    })
```

   Si ottiene il seguente risultato (nota che il suo punteggio non è ancora aumentato):

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

7. Puoi verificare che il suo punteggio sia cambiato con la seguente domanda:

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

   Si ottiene il seguente risultato:

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

Fase 8: Esplora

Complimenti! Hai completato con successo la Guida rapida ad Amazon DocumentDB.

Qual è il prossimo passo? Scopri come sfruttare appieno questo potente database con alcune delle sue funzionalità più popolari:

- Gestione di cluster e istanze Amazon DocumentDB
- Dimensionamento
- Backup e ripristino

> **Note**
>
> Per risparmiare sui costi, puoi interrompere il cluster Amazon DocumentDB per ridurre i costi o eliminare il cluster. Per impostazione predefinita, dopo 30 minuti di inattività, l' AWS Cloud9 ambiente interromperà l'istanza Amazon EC2 sottostante.

Connetti automaticamente un'istanza EC2 a un database Amazon DocumentDB esistente

La procedura seguente presuppone la presenza di un cluster Amazon DocumentDB e di un'istanza Amazon EC2 esistenti.

Accedi al tuo cluster Amazon DocumentDB e configura la connessione Amazon EC2

1. Accedi al tuo cluster Amazon DocumentDB.

    a. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb

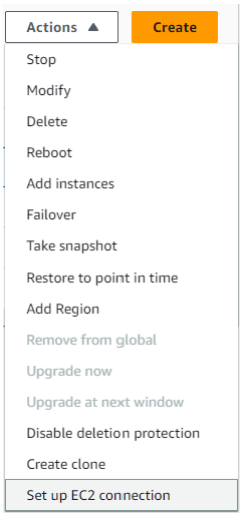
    b. Nel pannello di navigazione scegliere Cluster.

    > **Tip**
    >
    > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

    c. Specifica il cluster che desideri scegliendo il pulsante a sinistra del nome del cluster.

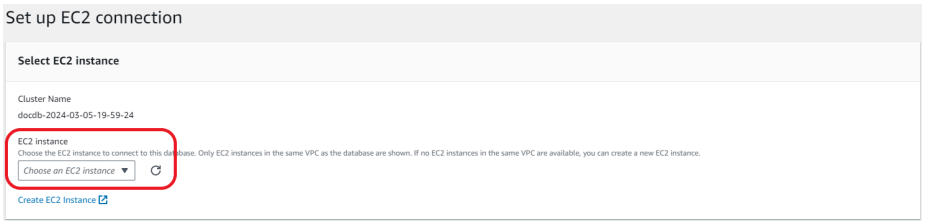
2. Configura la connessione Amazon EC2.

    a. Scegli Azioni, quindi scegli Configura connessione EC2.



Viene visualizzata la finestra di dialogo Configura connessione EC2.

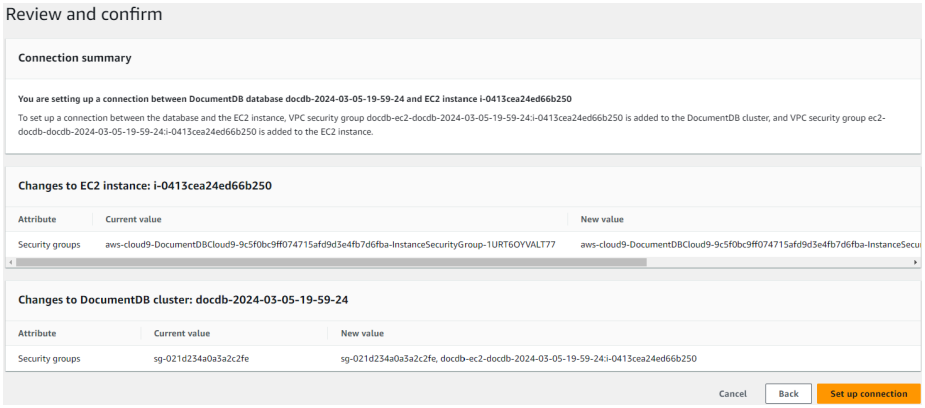
b. Nel campo dell'istanza EC2, scegli l'istanza EC2 che desideri connettere al tuo cluster.



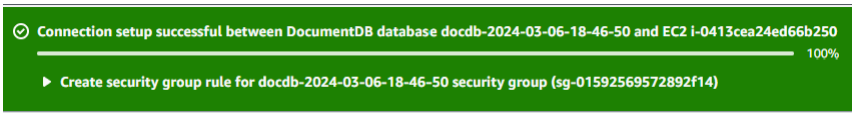
c. Scegli Continua.

Viene visualizzata la finestra di dialogo Rivedi e conferma.

d. Assicurati che le modifiche siano corrette. Quindi scegli Configura connessione.



In caso di esito positivo, viene visualizzata la seguente verifica:



Panoramica della connettività automatica con un'istanza EC2

Quando configuri una connessione tra un'istanza EC2 e un database Amazon DocumentDB, Amazon DocumentDB configura automaticamente il gruppo di sicurezza VPC per la tua istanza EC2 e per il tuo database Amazon DocumentDB.

Di seguito sono riportati i requisiti per connettere un'istanza EC2 a un database Amazon DocumentDB:

- L'istanza EC2 deve esistere nello stesso VPC del database Amazon DocumentDB.

  Se nello stesso VPC non esistono istanze EC2, allora la console fornisce un collegamento per crearne una.
- L'utente che configura la connettività deve disporre delle autorizzazioni per eseguire le seguenti operazioni Amazon EC2:
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeSecurityGroups`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

Se l'istanza database e l'istanza EC2 si trovano in zone di disponibilità diverse, è possibile che all'account vengano addebitati costi tra zone di disponibilità.

Quando si configura una connessione a un'istanza EC2, Amazon DocumentDB agisce in base alla configurazione corrente dei gruppi di sicurezza associati al database Amazon DocumentDB e all'istanza EC2, come descritto nella seguente tabella:

Configurazione attuale del gruppo di sicurezza Amazon DocumentDB	Configurazione del gruppo di sicurezza EC2 corrente	Azione Amazon DocumentDB
Esistono uno o più gruppi di sicurezza associati al database Amazon DocumentDB con un nome che corrisponde al modello. `DocumentDB-ec2-n` Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza include una sola regola in uscita con il gruppo di sicurezza VPC dell'istanza EC2 come origine.	Esistono uno o più gruppi di sicurezza associati all'istanza EC2 con un nome che corrisponde al modello `DocumentDB-ec2-n` (dove n è un numero). Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza ha solo una regola in uscita con il gruppo di sicurezza VPC del database Amazon DocumentDB come origine.	Amazon DocumentDB non intraprende alcuna azione. Una connessione era già configurata automaticamente tra l'istanza EC2 e il database Amazon DocumentDB. Poiché esiste già una connessione tra l'istanza EC2 e il database Amazon DocumentDB, i gruppi di sicurezza non vengono modificati.
Si applica una delle due condizioni seguenti: • Non esiste alcun gruppo di sicurezza associato al database Amazon DocumentDB con un nome che corrisponda al modello. `DocumentDB-ec2-n` • Esistono uno o più gruppi di sicurezza associati ad Amazon DocumentDB con un nome che corrisponde al modello. `DocumentDB-ec2-n` Tuttavia, Amazon DocumentDB non può utilizzare nessuno di questi gruppi di sicurezza per la connessione con l'istanza	Si applica una delle due condizioni seguenti: • Non esiste un gruppo di sicurezza associato all'istanza EC2 con un nome che corrisponde al modello `ec2-DocumentDB-n` . • Esistono uno o più gruppi di sicurezza associati all'istanza EC2 con un nome che corrisponde al modello `ec2-DocumentDB-n`. Tuttavia, Amazon DocumentDB non può utilizzare nessuno di questi gruppi di sicurezza per la connessione al database Amazon DocumentDB.	Azione Amazon DocumentDB: creare nuovi gruppi di sicurezza

Configurazione attuale del gruppo di sicurezza Amazon DocumentDB	Configurazione del gruppo di sicurezza EC2 corrente	Azione Amazon DocumentDB
EC2. Amazon DocumentDB non può utilizzare un gruppo di sicurezza che non dispone di una regola in entrata con il gruppo di sicurezza VPC dell'istanza EC2 come origine. Amazon DocumentDB inoltre non può utilizzare un gruppo di sicurezza che è stato modificato. Esempi di modifiche sono l'aggiunta di una regola o la modifica della porta di una regola esistente.	Amazon DocumentDB non può utilizzare un gruppo di sicurezza che non dispone di una regola in uscita con il gruppo di sicurezza VPC del database Amazon DocumentDB come origine. Amazon DocumentDB inoltre non può utilizzare un gruppo di sicurezza che è stato modificato.	

Configurazione attuale del gruppo di sicurezza Amazon DocumentDB	Configurazione del gruppo di sicurezza EC2 corrente	Azione Amazon DocumentDB
Esistono uno o più gruppi di sicurezza associati al database Amazon DocumentDB con un nome che corrisponde allo schema. `DocumentDB-ec2-n` Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza include una sola regola in uscita con il gruppo di sicurezza VPC dell'istanza EC2 come origine.	Esistono uno o più gruppi di sicurezza associati all'istanza EC2 con un nome che corrisponde al modello `ec2-DocumentDB-n` . Tuttavia, Amazon DocumentDB non può utilizzare nessuno di questi gruppi di sicurezza per la connessione al database Amazon DocumentDB. Amazon DocumentDB non può utilizzare un gruppo di sicurezza che non dispone di una regola in uscita con il gruppo di sicurezza VPC del database Amazon DocumentDB come origine. Amazon DocumentDB inoltre non può utilizzare un gruppo di sicurezza che è stato modificato.	Azione Amazon DocumentDB: creare nuovi gruppi di sicurezza

Configurazione attuale del gruppo di sicurezza Amazon DocumentDB	Configurazione del gruppo di sicurezza EC2 corrente	Azione Amazon DocumentDB
Esistono uno o più gruppi di sicurezza associati al database Amazon DocumentDB con un nome che corrisponde allo schema. `DocumentDB-ec2-n` Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza include una sola regola in uscita con il gruppo di sicurezza VPC dell'istanza EC2 come origine.	Esiste un gruppo di sicurezza EC2 valido per la connessione, ma non è associato all'istanza EC2. Questo gruppo di sicurezza ha un nome che corrisponde al modello `DocumentDB-ec2-n`. Non è stato modificato. Ha solo una regola in uscita con il gruppo di sicurezza VPC del database Amazon DocumentDB come origine.	Azione Amazon DocumentDB: associare il gruppo di sicurezza EC2

Configurazione attuale del gruppo di sicurezza Amazon DocumentDB	Configurazione del gruppo di sicurezza EC2 corrente	Azione Amazon DocumentDB
Si applica una delle due condizioni seguenti: • Non esiste alcun gruppo di sicurezza associato al database Amazon DocumentDB con un nome che corrisponda al modello. `DocumentDB-ec2-n` • Esistono uno o più gruppi di sicurezza associati al database Amazon DocumentDB con un nome che corrisponde al modello. `DocumentDB-ec2-n` Tuttavia, Amazon DocumentDB non può utilizzare nessuno di questi gruppi di sicurezza per la connessione con l'istanza EC2. Amazon DocumentDB non può utilizzare un gruppo di sicurezza che non dispone di una regola in entrata con il gruppo di sicurezza VPC dell'istanza EC2 come origine. Amazon DocumentDB inoltre non può utilizzare un gruppo di sicurezza che è stato modificato.	Esistono uno o più gruppi di sicurezza associati all'istanza EC2 con un nome che corrisponde al modello `DocumentDB-ec2-n` . Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza ha solo una regola in uscita con il gruppo di sicurezza VPC del database Amazon DocumentDB come origine.	Azione Amazon DocumentDB: creare nuovi gruppi di sicurezza

Azione Amazon DocumentDB: creare nuovi gruppi di sicurezza

Amazon DocumentDB intraprende le seguenti azioni:

- Crea un nuovo gruppo di sicurezza che corrisponde al modello `DocumentDB-ec2-n`. Questo gruppo di sicurezza include una regola in uscita con il gruppo di sicurezza VPC dell'istanza EC2 come origine. Questo gruppo di sicurezza è associato al database Amazon DocumentDB e consente all'istanza EC2 di accedere al database Amazon DocumentDB.
- Crea un nuovo gruppo di sicurezza che corrisponde al modello `ec2-DocumentDB-n`. Questo gruppo di sicurezza ha una regola in uscita con il gruppo di sicurezza VPC del database Amazon DocumentDB come origine. Questo gruppo di sicurezza è associato all'istanza EC2 e consente all'istanza EC2 di inviare traffico al database Amazon DocumentDB.

Azione Amazon DocumentDB: associare il gruppo di sicurezza EC2

Amazon DocumentDB associa il gruppo di sicurezza EC2 valido ed esistente all'istanza EC2. Questo gruppo di sicurezza consente all'istanza EC2 di inviare traffico al database Amazon DocumentDB.

Visualizzazione delle risorse di calcolo connesse

Puoi utilizzarlo Console di gestione AWS per visualizzare le risorse di calcolo connesse a un database Amazon DocumentDB. Le risorse mostrate includono le connessioni delle risorse di calcolo configurate automaticamente. È possibile configurare la connettività delle risorse di calcolo automaticamente nei modi seguenti:

- È possibile selezionare la risorsa di calcolo quando si crea il database. Per ulteriori informazioni, consulta Creazione di un cluster Amazon DocumentDB e Creazione di un cluster Multi-AZ DB.
- È possibile configurare la connettività tra un database esistente e una risorsa di calcolo. Per ulteriori informazioni, consulta Connetti Amazon EC2 automaticamente.

Le risorse di calcolo elencate non includono quelle connesse al database manualmente. Ad esempio, è possibile consentire manualmente a una risorsa di calcolo di accedere a un database aggiungendo una regola al gruppo di sicurezza VPC associato al database.

Per garantire la presenza della risorsa di calcolo nell'elenco, è necessario che siano soddisfatte le condizioni elencate di seguito.

- Il nome del gruppo di sicurezza associato alla risorsa di calcolo corrisponde al modello `ec2-DocumentDB-n` (dove n è un numero).

- Il gruppo di sicurezza associato alla risorsa di calcolo ha una regola in uscita con l'intervallo di porte impostato sulla porta utilizzata dal database Amazon DocumentDB.
- Il gruppo di sicurezza associato alla risorsa di calcolo ha una regola in uscita con l'origine impostata su un gruppo di sicurezza associato al database Amazon DocumentDB.
- Il nome del gruppo di sicurezza associato al database Amazon DocumentDB corrisponde allo schema `DocumentDB-ec2-n` (dove n è un numero).
- Il gruppo di sicurezza associato al database Amazon DocumentDB ha una regola in entrata con l'intervallo di porte impostato sulla porta utilizzata dal database Amazon DocumentDB.
- Il gruppo di sicurezza associato al database Amazon DocumentDB ha una regola in entrata con l'origine impostata su un gruppo di sicurezza associato alla risorsa di calcolo.

Per visualizzare le risorse di calcolo connesse a un database Amazon DocumentDB

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione, scegli Database, quindi scegli il nome del database Amazon DocumentDB.
3. Nella scheda Connettività e sicurezza, visualizza le risorse di calcolo nella sezione Risorse di calcolo connesse.

Connetti Amazon EC2 manualmente

Argomenti

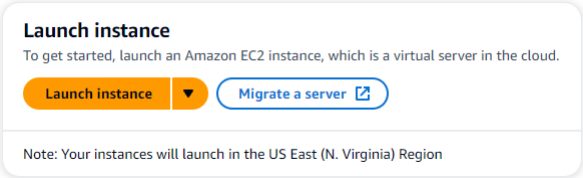
- Passaggio 1: creare un'istanza Amazon EC2
- Fase 2: creazione di un gruppo di sicurezza
- Passaggio 3: creare un cluster Amazon DocumentDB
- Passaggio 4: connettiti alla tua istanza Amazon EC2
- Passaggio 5: installa MongoDB Shell
- Passaggio 6: gestione di Amazon DocumentDB TLS
- Passaggio 7: connettiti al tuo cluster Amazon DocumentDB
- Fase 8: Inserimento e interrogazione dei dati
- Fase 9: Esplora

I passaggi seguenti presuppongono che tu abbia completato i passaggi indicati nell'Prerequisitiargomento.

Passaggio 1: creare un'istanza Amazon EC2

In questo passaggio, creerai un'istanza Amazon EC2 nella stessa regione e Amazon VPC che utilizzerai successivamente per il provisioning del tuo cluster Amazon DocumentDB.

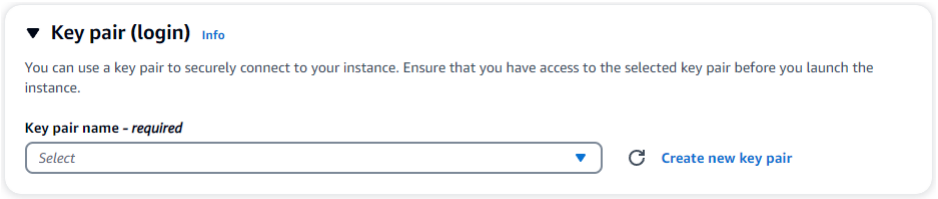
1. Sulla console Amazon EC2, scegli Launch instance.



2. Inserisci un nome o un identificatore nel campo Nome situato nella sezione Nome e tag.
3. Nell'elenco a discesa Amazon Machine Image (AMI), individua Amazon Linux 2 AMI e sceglila.



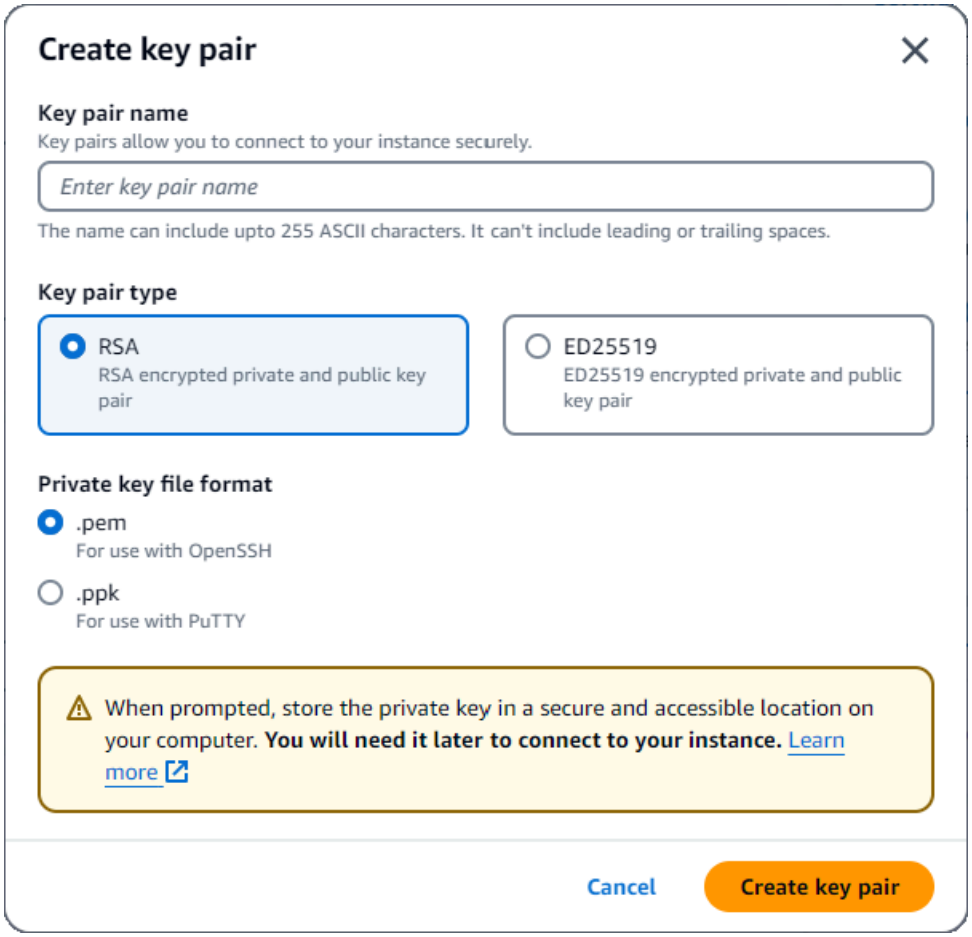
4. Individua e scegli t3.micro nell'elenco a discesa Tipo di istanza.
5. Nella sezione Coppia di chiavi (accesso), inserisci l'identificativo di una coppia di chiavi esistente o scegli Crea nuova coppia di chiavi.



È necessario fornire una coppia di chiavi Amazon EC2.

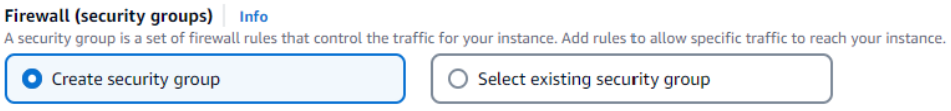
- Se disponi di una coppia di chiavi Amazon EC2:
  1. Seleziona una coppia di chiavi, scegline una dall'elenco.
  2. Devi avere già il file della chiave privata (.pem o .ppk) disponibile per accedere alla tua istanza Amazon EC2.
- Se non disponi di una coppia di chiavi Amazon EC2:
  1. Scegli Crea nuova coppia di chiavi, viene visualizzata la finestra di dialogo Crea coppia di chiavi.
  2. Inserite un nome nel campo Nome coppia di chiavi.

3. Scegliete il tipo di coppia di chiavi e il formato del file della chiave privata.
4. Scegliere Create key pair (Crea coppia di chiavi).



> **Note**
>
> Per motivi di sicurezza, consigliamo vivamente di utilizzare una coppia di chiavi per la connettività SSH e Internet all'istanza EC2.

6. Nella sezione Impostazioni di rete, in Firewall (gruppi di sicurezza), scegli Crea gruppo di sicurezza o Seleziona gruppo di sicurezza esistente.



Se hai scelto di selezionare un gruppo di sicurezza esistente, selezionane uno dall'elenco a discesa Gruppi di sicurezza comuni.

Se hai scelto di creare un nuovo gruppo di sicurezza, procedi come segue:

a. Controlla tutte le regole di autorizzazione del traffico che si applicano alla tua connettività EC2.

b. Nel campo IP, scegli Il mio IP o seleziona Personalizzato per scegliere da un elenco di blocchi CIDR, elenchi di prefissi o gruppi di sicurezza. Non consigliamo Anywhere come opzione, a meno che l'istanza EC2 non si trovi su una rete isolata, poiché consente l'accesso all'istanza EC2 con qualsiasi indirizzo IP.

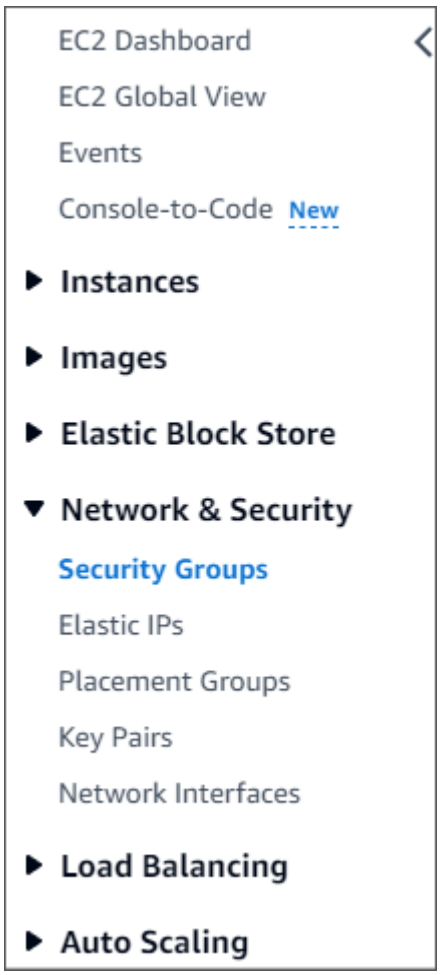


7. Nella sezione Riepilogo, rivedi la tua configurazione EC2 e scegli Launch instance, se corretta.

Fase 2: creazione di un gruppo di sicurezza

Ora creerai un nuovo gruppo di sicurezza nel tuo Amazon VPC predefinito. Il gruppo di sicurezza ti `demoDocDB` consente di connetterti al tuo cluster Amazon DocumentDB sulla porta 27017 (la porta predefinita per Amazon DocumentDB) dalla tua istanza Amazon EC2.

1. Nella Console di gestione Amazon EC2, in Rete e sicurezza, scegli Gruppi di sicurezza.



2. Scegliere Create Security Group (Crea gruppo di sicurezza).



3. Nella sezione Dettagli di base:

   a. In Security group name (Nome gruppo di sicurezza) immettere `demoDocDB`.

   b. In Description (Descrizione), inserire una descrizione.

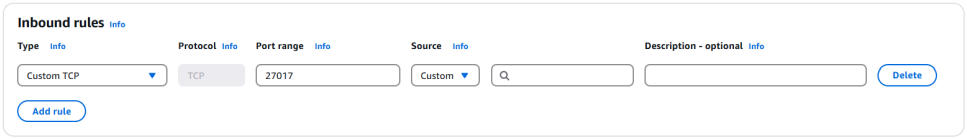
   c. Per il VPC, accetta l'utilizzo del tuo VPC predefinito.

4. Nella sezione Regole in entrata, scegliere Aggiungi regola.

   a. Per Tipo, scegli Regola TCP personalizzata (impostazione predefinita).

   b. Per Intervallo di porte, inserisci `27017`.

   c. In Source (Origine), scegliere Custom (Personalizzata). Nel campo accanto, cerca il gruppo di sicurezza che hai appena creato nel passaggio 1. Potrebbe essere necessario aggiornare il browser affinché la console Amazon EC2 compili automaticamente il nome della fonte.



5. Accetta tutte le altre impostazioni predefinite e scegli Crea gruppo di sicurezza.



Passaggio 3: creare un cluster Amazon DocumentDB

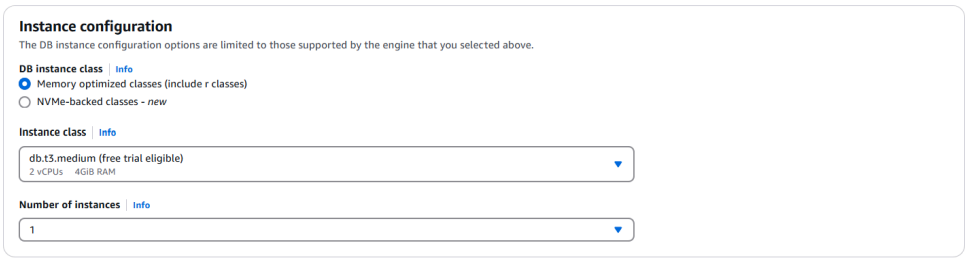
Durante il provisioning dell'istanza Amazon EC2, creerai il tuo cluster Amazon DocumentDB.

1. Accedi alla console Amazon DocumentDB e scegli Clusters dal pannello di navigazione.
2. Scegli Crea.
3. Lascia l'impostazione del tipo di cluster come predefinita, Instance Based Cluster.
4. Nella configurazione del cluster, per l'identificatore del cluster, inserisci un nome univoco. Tieni presente che la console cambierà tutti i nomi dei cluster in lettere minuscole indipendentemente da come vengono immessi.

   Lasciare la versione del motore al valore predefinito di 5.0.0.
5. Per la configurazione dello storage in cluster, lascia l'impostazione predefinita di Amazon DocumentDB Standard.
6. Nella configurazione dell'istanza:

   - Per la classe di istanza DB, scegli le classi ottimizzate per la memoria (includi le classi r) (questa è l'impostazione predefinita).

     L'altra opzione di istanza sono NVMe-backed le classi. Per ulteriori informazioni, consulta NVMe-backed istanze.
   - Per la classe Instance, scegli il tipo di istanza più adatto alle tue esigenze. Per una spiegazione più dettagliata delle classi di istanze, consultaSpecifiche della classe di istanza.
   - Per il numero di istanze, scegliete un numero che rispecchi al meglio le vostre esigenze. Ricorda che più basso è il numero, minore è il costo e minore è il read/write volume che può essere gestito dal cluster.



7. Per la connettività, lascia l'impostazione predefinita di Non connetterti a una risorsa di calcolo EC2.

> **Note**
>
> La connessione a una risorsa di calcolo EC2 crea automaticamente gruppi di sicurezza per la connessione al cluster. Poiché hai creato manualmente questi gruppi di sicurezza nel passaggio precedente, dovresti selezionare Non connetterti a una risorsa di calcolo EC2 per non creare un secondo set di gruppi di sicurezza.

8. Nella sezione Autenticazione, inserisci un nome utente per l'utente principale, quindi scegli Gestito autonomamente. Inserisci una password, quindi confermala.

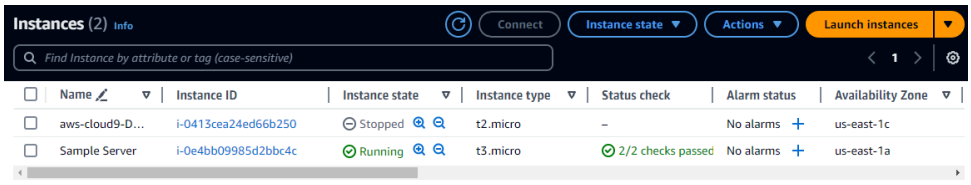
   Se invece hai scelto Managed in AWS Secrets Manager, consulta Gestione delle password con Amazon DocumentDB e AWS Secrets Manager per ulteriori informazioni.

9. Scegli Crea cluster.

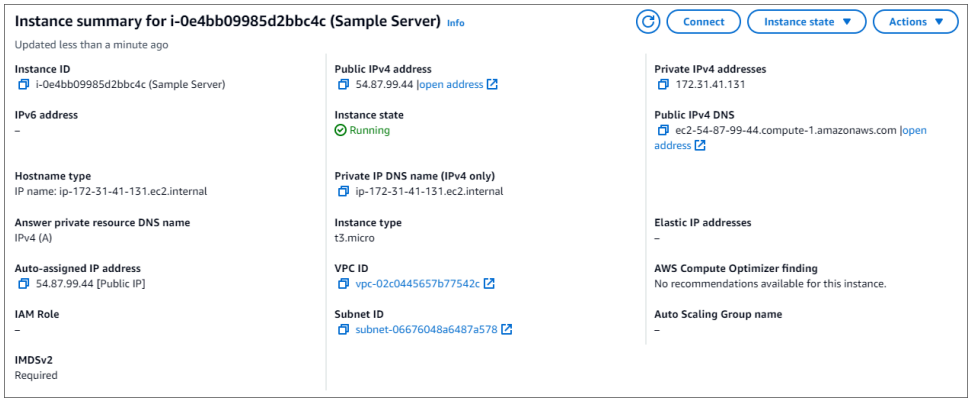
Passaggio 4: connettiti alla tua istanza Amazon EC2

La connessione alla tua istanza Amazon EC2 ti consentirà di installare la shell MongoDB. L'installazione di mongo shell ti consente di connetterti e interrogare il tuo cluster Amazon DocumentDB. Completa questa procedura:

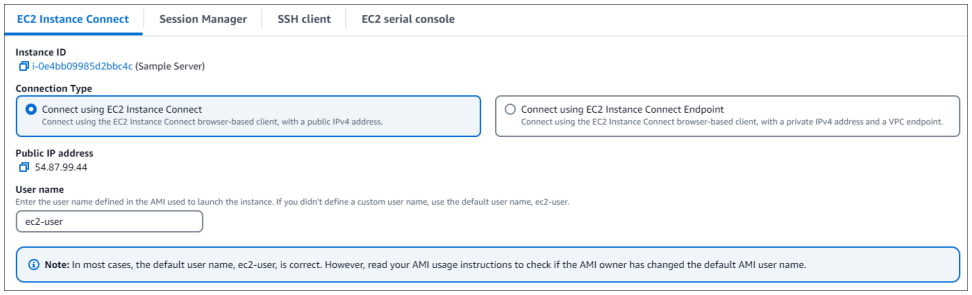
1. Sulla console Amazon EC2, accedi alle tue istanze e verifica se l'istanza appena creata è in esecuzione. In tal caso, seleziona l'istanza facendo clic sull'ID dell'istanza.



2. Scegli Connetti.



3. Sono disponibili quattro opzioni a schede per il metodo di connessione: Amazon EC2 Instance Connect, Session Manager, client SSH o console seriale EC2. Devi sceglierne una e seguirne le istruzioni. Al termine, scegli Connetti.



> **Note**
>
> Se il tuo indirizzo IP è cambiato dopo aver iniziato questa procedura dettagliata o tornerai al tuo ambiente in un secondo momento, devi aggiornare la regola in entrata del tuo gruppo di demoEC2 sicurezza per abilitare il traffico in entrata dal tuo nuovo indirizzo API.

Passaggio 5: installa MongoDB Shell

Ora puoi installare la shell MongoDB, un'utilità a riga di comando che usi per connetterti e interrogare il tuo cluster Amazon DocumentDB. Attualmente esistono due versioni di MongoDB shell: la versione più recente, mongosh, e la versione precedente, mongo shell.

> **Important**
>
> Esiste una limitazione nota con Node.js i driver precedenti alla versione 6.13.1, che attualmente non sono supportati dall'autenticazione dell'identità IAM per Amazon

DocumentDB. Node.js i driver e gli strumenti che utilizzano Node.js driver (ad esempio, mongosh) devono essere aggiornati per utilizzare la versione 6.13.1 o successiva. Node.js

Segui le istruzioni seguenti per installare la shell MongoDB per il tuo sistema operativo.

On Amazon Linux

Per installare la shell MongoDB su Amazon Linux

Se non utilizzi IAM e desideri utilizzare la shell MongoDB più recente (mongosh) per connetterti al tuo cluster Amazon DocumentDB, segui questi passaggi:

1. Crea il file del repository. Nella riga di comando dell'istanza EC2 che hai creato, esegui il seguente comando:

```
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. Al termine, installa mongosh con una delle due seguenti opzioni di comando al prompt dei comandi:

   Opzione 1: se hai scelto Amazon Linux 2023 predefinito durante la configurazione di Amazon EC2, inserisci questo comando:

   ```
   sudo yum install -y mongodb-mongosh-shared-openssl3
   ```

   Opzione 2: se hai scelto Amazon Linux 2 durante la configurazione di Amazon EC2, inserisci questo comando:

   ```
   sudo yum install -y mongodb-mongosh
   ```

Se utilizzi IAM, devi utilizzare la versione precedente della shell MongoDB (5.0) per connetterti al tuo cluster Amazon DocumentDB, segui questi passaggi:

1. Crea il file del repository. Nella riga di comando dell'istanza EC2 che hai creato, esegui il seguente comando:

```
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. Al termine, installa la shell mongodb 5.0 con la seguente opzione di comando al prompt dei comandi:

```
sudo yum install -y mongodb-org-shell
```

On Ubuntu

Per installare mongosh su Ubuntu

1. Importa la chiave pubblica che verrà utilizzata dal sistema di gestione dei pacchetti.

```
curl -fsSL https://pgp.mongodb.com/server-5.0.asc | sudo gpg --dearmor -o /usr/
share/keyrings/mongodb-server-5.0.gpg
```

2. Crea l'elenco di file `mongodb-org-5.0.list` per MongoDB utilizzando il comando appropriato per la versione di Ubuntu.

```
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. Importa e aggiorna il database locale dei pacchetti usando il seguente comando:

```
sudo apt-get update
```

4. Installa mongosh.

```
sudo apt-get install -y mongodb-mongosh
```

Per informazioni sull'installazione di versioni precedenti di MongoDB nel sistema Ubuntu, consulta Installazione di MongoDB Community Edition in Ubuntu.

On other operating systems

Per installare la shell Mongo in altri sistemi operativi, vedi l'argomento relativo all'installazione di MongoDB Community Edition nella documentazione relativa a MongoDB.

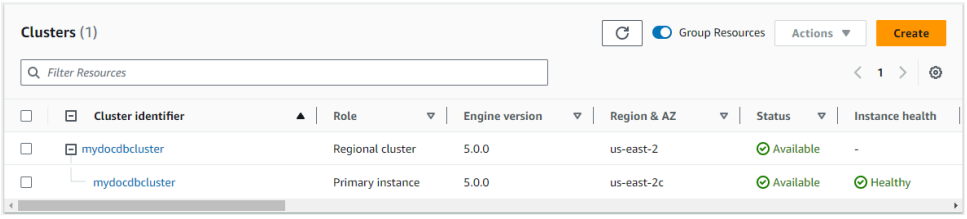
Passaggio 6: gestione di Amazon DocumentDB TLS

Scarica il certificato CA per Amazon DocumentDB con il codice seguente: `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

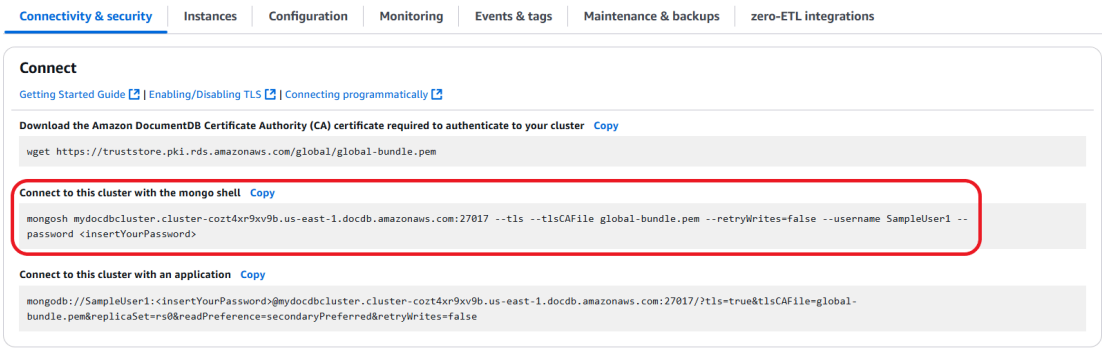
> **Note**
>
> Transport Layer Security (TLS) è abilitato di default per tutti i nuovi cluster Amazon DocumentDB. Per ulteriori informazioni, consulta Crittografia dei dati in transito.

Passaggio 7: connettiti al tuo cluster Amazon DocumentDB

1. Nella console Amazon DocumentDB, in Clusters, individua il cluster. Scegli il cluster che hai creato facendo clic sull'identificatore del cluster per quel cluster.



2. Nella scheda Connettività e sicurezza, individua Connetti a questo cluster con mongo shell nella casella Connect:



Copia la stringa di connessione fornita e incollala nel tuo terminale.

Apporta le seguenti modifiche:

a. Assicurati di avere il nome utente corretto nella stringa.

b. Ometti `<insertYourPassword>` in modo che ti venga richiesta la password dalla shell mongo quando ti connetti.

c. Facoltativo: se utilizzi l'autenticazione IAM o stai utilizzando la versione precedente della shell MongoDB, modifica la stringa di connessione come segue:

```
mongo --ssl --host mydocdbcluster.cluster-cozt4xr9xv9b.us-
east-1.docdb.amazonaws.com:27017 --sslCAFile global-bundle.pem --
username SampleUser1 --password
```

Sostituisci `mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1` con le stesse informazioni del tuo cluster.

3. Premi invio nel tuo terminale. Ora ti verrà richiesta la password. Inserisci la password.
4. Quando inserisci la password e viene visualizzata la `rs0 [direct: primary] <env-name>>` richiesta, sei connesso correttamente al tuo cluster Amazon DocumentDB.

Per risolvere i problemi di connessione, consulta[Risoluzione dei problemi di Amazon DocumentDB](#).

Fase 8: Inserimento e interrogazione dei dati

Ora che sei connesso al tuo cluster, puoi eseguire alcune query per acquisire familiarità con l'utilizzo di un database di documenti.

1. Per inserire un singolo documento, inserisci quanto segue:

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

Si ottiene il seguente risultato:

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

2. Puoi leggere il documento che hai scritto con il `findOne()` comando (perché restituisce solo un singolo documento). Inserisci quanto segue:

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

Si ottiene il seguente risultato:

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

3. Per eseguire qualche altra domanda, considera un caso d'uso dei profili di gioco. Innanzitutto, inserisci alcune voci in una raccolta intitolata`profiles`. Inserisci quanto segue:

```
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 }
])
```

Si ottiene il seguente risultato:

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

4. Usa il `find()` comando per restituire tutti i documenti nella raccolta dei profili. Inserisci quanto segue:

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

Otterrai un risultato che corrisponderà ai dati che hai digitato nel passaggio 3.

5. Usa una query per un singolo documento usando un filtro. Inserisci quanto segue:

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

Si ottiene il seguente risultato:

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

6. Ora proviamo a trovare un profilo e modificarlo usando il `findAndModify` comando. Daremo all'utente Matt altri 10 punti con il seguente codice:

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

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

Si ottiene il seguente risultato (nota che il suo punteggio non è ancora aumentato):

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

7. Puoi verificare che il suo punteggio sia cambiato con la seguente domanda:

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

   Si ottiene il seguente risultato:

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

Fase 9: Esplora

Complimenti! Hai completato con successo la Guida rapida ad Amazon DocumentDB.

Qual è il prossimo passo? Scopri come sfruttare appieno questo potente database con alcune delle sue funzionalità più popolari:

- Gestione di cluster e istanze Amazon DocumentDB
- Dimensionamento
- Backup e ripristino

> **Note**
>
> Per risparmiare sui costi, puoi interrompere il cluster Amazon DocumentDB per ridurre i costi o eliminare il cluster. Per impostazione predefinita, dopo 30 minuti di inattività, l' AWS Cloud9 ambiente interromperà l'istanza Amazon EC2 sottostante.

Connettiti utilizzando il driver JDBC di Amazon DocumentDB

Il driver JDBC per Amazon DocumentDB fornisce un' SQL-relational interfaccia per gli sviluppatori e consente la connettività da strumenti di BI come Tableau e. DbVisualizer

Per informazioni più dettagliate, consulta la documentazione del driver JDBC di Amazon DocumentDB su. GitHub

Argomenti

- Nozioni di base
- Connettiti ad Amazon DocumentDB da Tableau Desktop
- Connettiti ad Amazon DocumentDB da DbVisualizer
- Generazione automatica di schemi JDBC
- Supporto e limitazioni SQL
- Risoluzione dei problemi

Nozioni di base

Passaggio 1. Crea un cluster Amazon DocumentDB

Se non hai creato un cluster Amazon DocumentDB, creane uno utilizzando le istruzioni in. Inizia a usare Amazon DocumentDB

> **Note**
>
> Amazon DocumentDB è un servizio esclusivamente di Virtual Private Cloud (VPC). Se ti connetti da una macchina locale, esterna al VPC del cluster, dovrai creare una connessione SSH a un'istanza Amazon EC2. In questo caso, avvia il cluster utilizzando le istruzioni in. Connettiti tramite Amazon EC2 Per ulteriori informazioni sul tunneling SSH e su quando potresti averne bisogno, consulta Using an SSH Tunneling to Connect to Amazon DocumentDB.

Passaggio 2. Installazione JRE o JDK

A seconda dell'applicazione BI, potrebbe essere necessario assicurarsi che sul computer sia installata una versione 8 o successiva di JRE o JDK a 64 bit. Puoi scaricare Java SE Runtime Environment 8 qui. https://www.oracle.com/ca-en/java/technologies/javase-jre8-downloads.html

Fase 3. Scarica il driver JDBC DocumentDB

Scarica il driver JDBC DocumentDB da qui. https://github.com/aws/amazon-documentdb-jdbc-driver/releases Il driver è confezionato in un unico file JAR (ad esempio documentdb-jdbc-1.0.0-all.jar).

Passaggio 4. Utilizzo di un tunnel SSH per connettersi ad Amazon DocumentDB

I cluster Amazon DocumentDB (con compatibilità con MongoDB) vengono distribuiti all'interno di un Amazon Virtual Private Cloud (Amazon VPC). È possibile accedervi direttamente dalle istanze Amazon EC2 o da altri AWS servizi distribuiti nello stesso Amazon VPC. Inoltre, è possibile accedere ad Amazon DocumentDB da istanze EC2a o altri AWS servizi in diversi VPC nella stessa regione o in altre regioni tramite peering VPC. AWS

Puoi utilizzare il tunneling SSH (noto anche come port forwarding) per accedere alle tue risorse Amazon DocumentDB, dall'esterno del VPC del cluster. Questo accadrà per la maggior parte degli utenti che non eseguono la propria applicazione su una macchina virtuale nello stesso VPC del cluster DocumentDB.

Per creare un tunnel SSH, è necessaria un'istanza Amazon EC2 in esecuzione nello stesso Amazon VPC del cluster Amazon DocumentDB. Puoi utilizzare un'istanza EC2 esistente nello stesso VPC del cluster o crearne una. Puoi configurare un tunnel SSH verso il cluster Amazon DocumentDB `sample-cluster.node.us-east-1.docdb.amazonaws.com` eseguendo il seguente comando sul tuo computer locale.

```
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
```

Il flag -L viene utilizzato per inoltrare una porta locale. Questo è un prerequisito per la connessione a qualsiasi strumento di BI in esecuzione su un client esterno al tuo VPC. Una volta eseguito il passaggio precedente, puoi passare ai passaggi successivi per lo strumento di BI di tua scelta.

Per ulteriori informazioni sul tunneling SSH, consulta la documentazione sull'uso di un tunnel SSH per connettersi ad Amazon DocumentDB.

Connettiti ad Amazon DocumentDB da Tableau Desktop

Argomenti

- Aggiungere il driver JDBC Amazon DocumentDB
- Connessione ad Amazon DocumentDB tramite Tableau - SSH Tunnel

Aggiungere il driver JDBC Amazon DocumentDB

Per connetterti ad Amazon DocumentDB da Tableau Desktop, devi scaricare e installare il driver JDBC Amazon DocumentDB e il connettore DocumentDB Tableau.

1. Scarica il file JAR del driver JDBC di Amazon DocumentDB dal repository Amazon DocumentDB JDBC Driver e copialo in una di queste directory in base al tuo sistema operativo:
   - Windows - `C:\Program Files\Tableau\Drivers`
   - macOS - `~/Library/Tableau/Drivers`
2. Scarica il connettore DocumentDB Tableau (un file TACO) dal sito Web di Tableau Exchange e copialo nella tua directory My Tableau. Repository/Connectors
   - Windows - `C:\Users\[user]\Documents\My Tableau Repository\Connectors`
   - macOS - `/Users/[user]/Documents/My Tableau Repository/Connectors`

Per ulteriori informazioni, consulta la documentazione di Tableau.

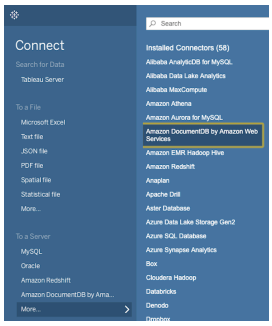
> **Note**
>
> Se utilizzi certificati CA più recenti, assicurati di aggiornare il driver JDBC alla versione 1.4.5 (disponibile in questo repository). AWS GitHub

Connessione ad Amazon DocumentDB tramite Tableau - SSH Tunnel

Per connetterti a Tableau da un computer client esterno al VPC del cluster DocumentDB, devi configurare un tunnel SSH prima di seguire i passaggi seguenti:

1. Avvia l'applicazione Tableau Desktop.
2. Vai a Connetti > A un server > Altro.

3. Scegli Amazon DocumentDB di Amazon Web Services in Connettori installati.

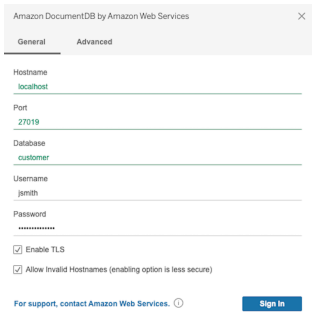


Connessione ad Amazon DocumentDB tramite Tableau - tunnel SSH esterno

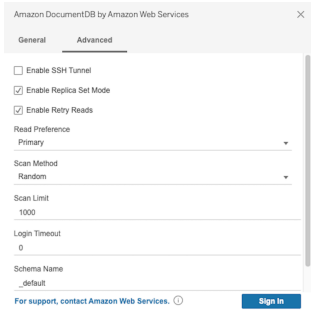
1. Inserisci i parametri di connessione richiesti Nome host, Porta, Database , Nome utente e Password. I parametri di connessione nell'esempio seguente sono equivalenti alla stringa di connessione JDBC:

   ```
   jdbc:documentdb://localhost:27019/test?
   tls=true&tlsAllowInvalidHostnames=true&scanMethod=random&scanLimit=1000&logir
   ```

   i parametri nome utente e password passati separatamente in una raccolta di proprietà. Per ulteriori informazioni sui parametri delle stringhe di connessione, consulta la documentazione github del driver JDBC di Amazon DocumentDB.



2. (Facoltativo) Altre opzioni avanzate sono disponibili nella scheda Avanzate.

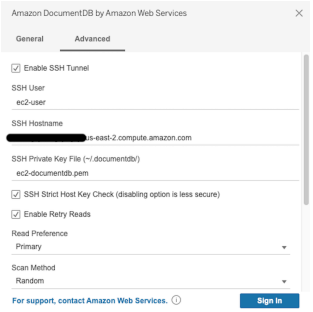


3. Selezionare Sign in (Accedi).

Connessione ad Amazon DocumentDB tramite Tableau: tunnel SSH interno

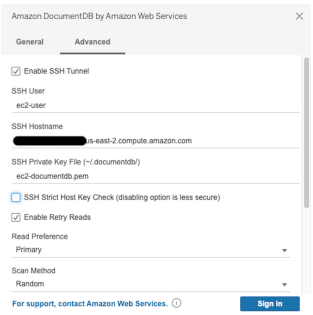
> **Note**
>
> Se preferisci non configurare il tunnel SSH utilizzando un terminale, puoi utilizzare la GUI di Tableau per specificare i dettagli dell'istanza EC2 che il driver JDBC utilizzerà intrinsecamente per creare un tunnel SSH.

1. Nella scheda Avanzate, scegli l'opzione Abilita tunnel SSH per esaminare ulteriori proprietà.



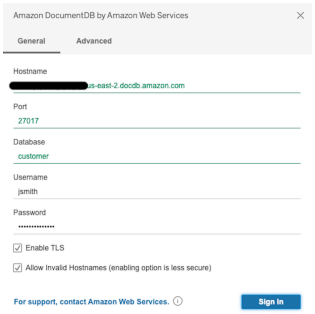
2. Inserisci l'utente SSH, il nome host SSH e il file della chiave privata SSH.
3. (Facoltativo) È possibile disattivare l'opzione SSH Strict Host Key Check che ignora il controllo della chiave host rispetto a un file hosts noto.

> **Note**
>
> La disattivazione di questa opzione è meno sicura in quanto può portare a un attacco man-in-the-middle.



4. Inserisci i parametri richiesti: nome host, porta, database , nome utente e password.

> **Note**
>
> Assicurati di utilizzare l'endpoint del cluster DocumentDB e non localhost quando utilizzi l'opzione del tunnel SSH interno.



5. Selezionare Sign in (Accedi).

Connettiti ad Amazon DocumentDB da DbVisualizer

Argomenti

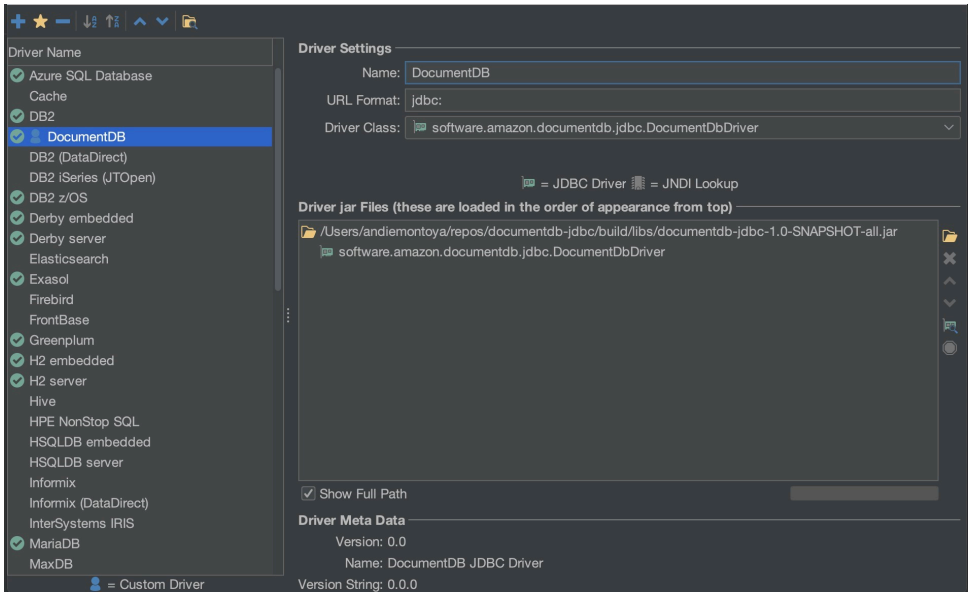
- Aggiungere il driver JDBC Amazon DocumentDB
- Connessione ad Amazon DocumentDB utilizzando DbVisualizer

Aggiungere il driver JDBC Amazon DocumentDB

Per connetterti ad Amazon DocumentDB da, DbVisualizer devi prima importare il driver JDBC di Amazon DocumentDB

1. Avvia l' DbVisualizer applicazione e vai al percorso del menu: Strumenti > Driver Manager...
2. Scegli + (o nel menu, seleziona Autista > Crea autista).
3. Impostare Name (Nome) su `DocumentDB`.
4. Imposta il formato URL su `jdbc:documentdb://<host>[:port]/<database>[?option=value[&option=value[...]]]`
5. Scegli il pulsante della cartella, quindi seleziona il file JAR del driver JDBC di Amazon DocumentDB e scegli il pulsante Apri.

6. Verifica che il campo Driver Class sia impostato su. `software.amazon.documentdb.jdbc.DocumentDbDriver` Le impostazioni di Driver Manager per DocumentDB dovrebbero essere simili all'esempio seguente.



7. Chiudere la finestra di dialogo. Il driver JDBC di Amazon DocumentDB sarà configurato e pronto per l'uso.

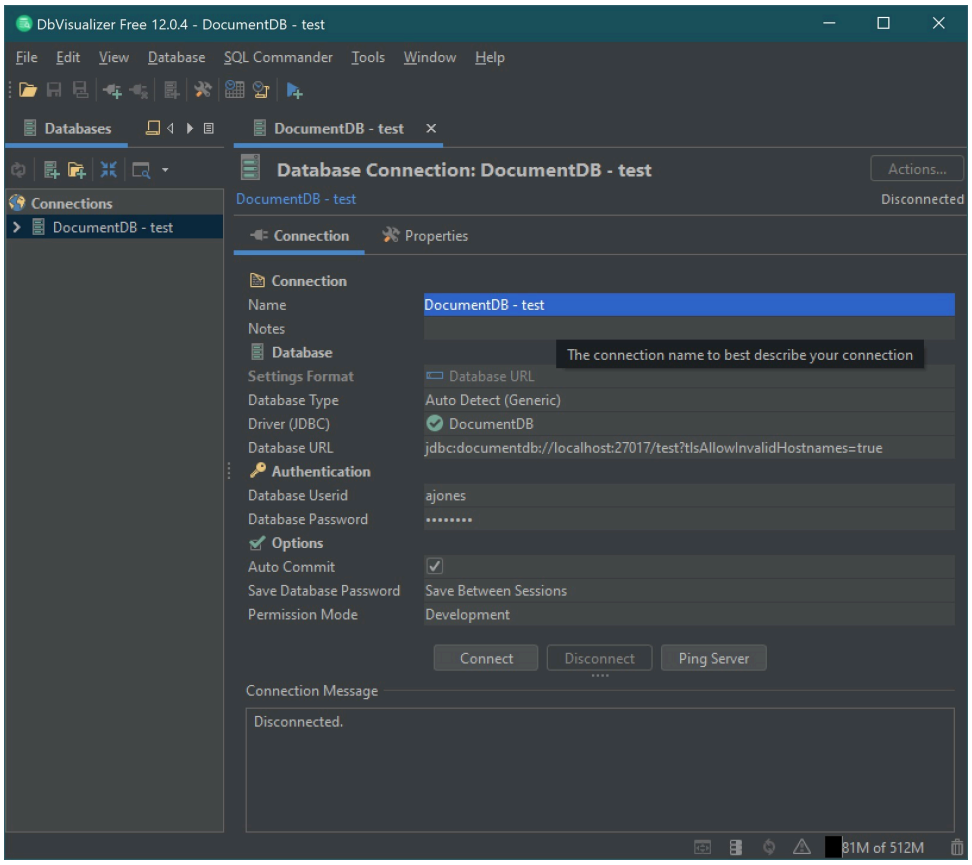
Connessione ad Amazon DocumentDB utilizzando DbVisualizer

Connettiti ad Amazon DocumentDB utilizzando DbVisualizer

1. Se ti connetti dall'esterno del VPC del cluster Amazon DocumentDB, assicurati di aver configurato un tunnel SSH.
2. Scegli Database > Crea connessione al database dal menu di livello superiore.
3. Immettete un nome descrittivo per il campo Nome.
4. Imposta Driver (JDBC) sul driver DocumentDB creato nella sezione precedente.
5. Imposta l'URL del database sulla stringa di connessione JDBC.

   Ad esempio: `jdbc:documentdb://localhost:27017/database?tlsAllowInvalidHostnames=true`
6. Imposta Database Userid sul tuo ID utente Amazon DocumentDB.
7. Imposta la password del database sulla password corrispondente per l'ID utente.

La finestra di dialogo di connessione al database dovrebbe assomigliare alla seguente finestra di dialogo:



8. Scegli Connetti.

Generazione automatica di schemi JDBC

Amazon DocumentDB è un database di documenti e pertanto non ha il concetto di tabelle e schema. Tuttavia, gli strumenti di BI come Tableau si aspettano che il database a cui si connette presenti uno schema. In particolare, quando la connessione al driver JDBC deve ottenere lo schema per la raccolta nel database, eseguirà il polling di tutte le raccolte nel database. Il driver determinerà se esiste già una versione memorizzata nella cache dello schema per quella raccolta. Se non esiste una versione memorizzata nella cache, campionerà la raccolta per i documenti e creerà uno schema basato sul seguente comportamento.

Argomenti

- Limitazioni alla generazione dello schema
- Opzioni del metodo di scansione

- Tipi di dati Amazon DocumentDB
- Mappatura dei campi scalari del documento
- Gestione dei tipi di dati di oggetti e array

Limitazioni alla generazione dello schema

Il driver JDBC DocumentDB impone un limite alla lunghezza degli identificatori a 128 caratteri. Il generatore di schemi può troncare la lunghezza degli identificatori generati (nomi di tabelle e nomi di colonne) per garantire che soddisfino tale limite.

Opzioni del metodo di scansione

Il comportamento di campionamento può essere modificato utilizzando le opzioni della stringa di connessione o dell'origine dati.

- Metodo di scansione = <option>
  - random - (impostazione predefinita) - I documenti di esempio vengono restituiti in ordine casuale.
  - idForward: i documenti di esempio vengono restituiti in ordine di id.
  - idReverse: i documenti di esempio vengono restituiti in ordine inverso all'id.
  - all: campiona tutti i documenti della raccolta.
- scanLimit= <n>- Il numero di documenti da campionare. Il valore deve essere un numero intero positivo. Il valore predefinito è 1000. Se ScanMethod è impostato su all, questa opzione viene ignorata.

Tipi di dati Amazon DocumentDB

Il server Amazon DocumentDB supporta diversi tipi di dati MongoDB. Di seguito sono elencati i tipi di dati supportati e i tipi di dati JDBC associati.

Tipo di dati MongoDB	Supportato in DocumentDB	Tipo di dati JDBC
Dati binari	Sì	VARBINARY
Booleano	Sì	BOOLEAN
Double	Sì	DOUBLE

Tipo di dati MongoDB	Supportato in DocumentDB	Tipo di dati JDBC
Numero intero a 32 bit	Sì	INTEGER
Numero intero a 64 bit	Sì	BIGINT
Stringa	Sì	VARCHAR
ObjectId	Sì	VARCHAR
Data	Sì	TIMESTAMP
Null	Sì	VARCHAR
Espressione regolare	Sì	VARCHAR
Time stamp	Sì	VARCHAR
MinKey	Sì	VARCHAR
MaxKey	Sì	VARCHAR
Oggetto	Sì	tavolo virtuale
Array	Sì	tavolo virtuale
Decimal128	No	DECIMAL
JavaScript	No	VARCHAR
JavaScript (con ambito)	No	VARCHAR
Undefined	No	VARCHAR
Symbol	No	VARCHAR
dBPointer (4.0+)	No	VARCHAR

Mappatura dei campi scalari del documento

Durante la scansione di un campione di documenti da una raccolta, il driver JDBC creerà uno o più schemi per rappresentare gli esempi della raccolta. In generale, un campo scalare nel documento è mappato su una colonna dello schema della tabella. Ad esempio, in una raccolta denominata team e in un singolo documento`{ "_id" : "112233", "name" : "Alastair", "age": 25 }`, questo dovrebbe corrispondere allo schema:

Nome tabella	Nome colonna	Tipo di dati	Chiave
team	id del team	VARCHAR	PK
team	nome	VARCHAR	
team	età	INTEGER	

Promozione dei conflitti tra tipi di dati

Durante la scansione dei documenti campione, è possibile che i tipi di dati per un campo non siano coerenti da un documento all'altro. In questo caso, il driver JDBC promuoverà il tipo di dati JDBC a un tipo di dati comune adatto a tutti i tipi di dati dei documenti campionati.

Ad esempio:

```
{
"_id" : "112233",
"name" : "Alastair", "age" : 25
}

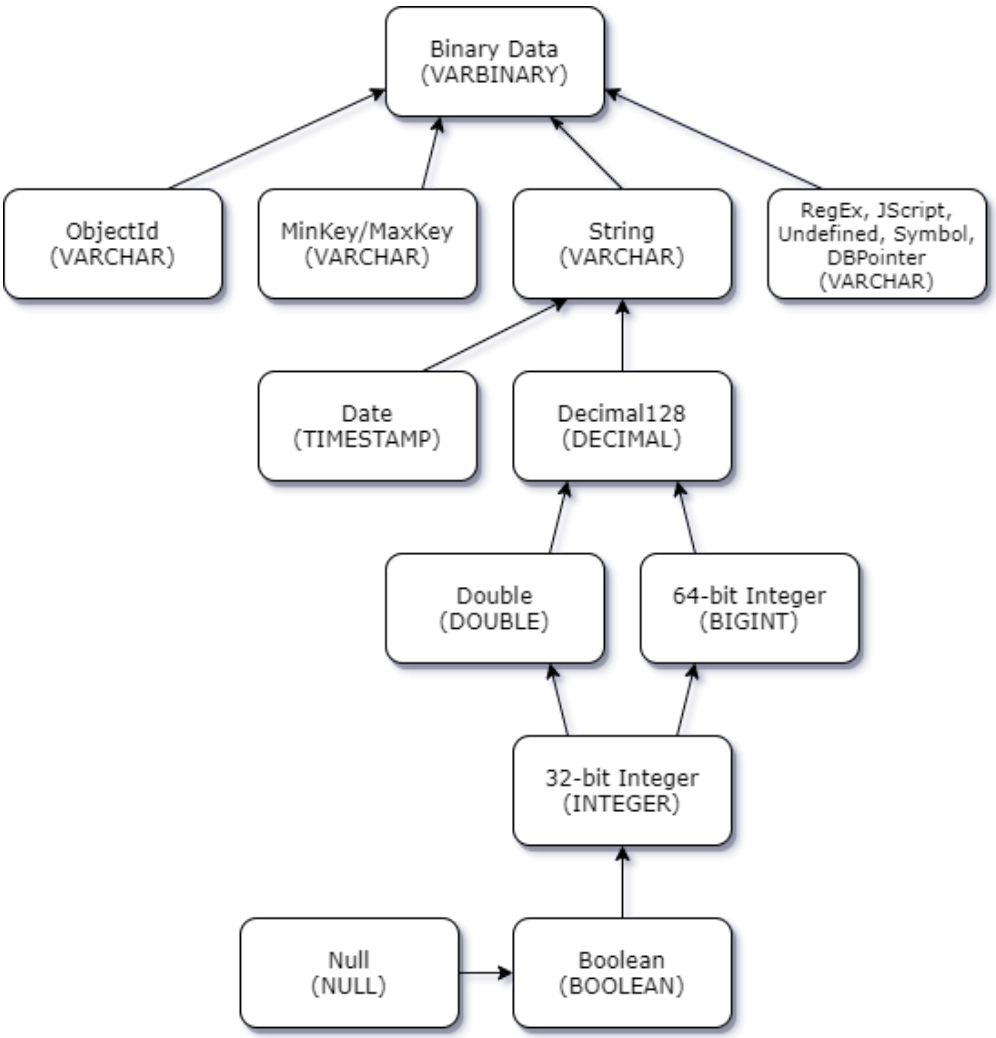
{
"_id" : "112244",
"name" : "Benjamin",
"age" : "32"
}
```

Il campo dell'età è di tipo intero a 32 bit nel primo documento ma stringa nel secondo documento. Qui il driver JDBC promuoverà il tipo di dati JDBC a VARCHAR per gestire entrambi i tipi di dati quando vengono rilevati.

Nome tabella	Nome colonna	Tipo di dati	Chiave
team	ID squadra	VARCHAR	PK
team	nome	VARCHAR	
team	età	VARCHAR	

Scalar-scalar promozione dei conflitti

Il diagramma seguente mostra il modo in cui vengono risolti i conflitti tra tipi di dati scalari.



Scalar-complex promozione dei conflitti di tipo

Analogamente ai conflitti di tipo scalare-scalare, lo stesso campo in documenti diversi può avere tipi di dati in conflitto tra complessi (array e oggetto) e scalari (intero, booleano, ecc.). Tutti questi conflitti vengono risolti (promossi) a VARCHAR per quei campi. In questo caso, i dati dell'array e dell'oggetto vengono restituiti come rappresentazione JSON.

Embedded Array - Esempio di conflitto tra campi di stringhe:

```
{
   "_id":"112233",
   "name":"George Jackson",
   "subscriptions":[
      "Vogue",
      "People",
      "USA Today"
   ]
}
{
   "_id":"112244",
   "name":"Joan Starr",
   "subscriptions":1
}
```

L'esempio precedente è mappato allo schema della tabella customer2:

Nome tabella	Nome colonna	Tipo di dati	Chiave
cliente2	id cliente2	VARCHAR	PK
cliente 2	nome	VARCHAR	
cliente 2	sottoscrizione	VARCHAR	

e la tabella virtuale customer1_subscriptions:

Nome tabella	Nome colonna	Tipo di dati	Chiave
customer1_subscriptions	ID cliente1	VARCHAR	PK/FK

Nome tabella	Nome colonna	Tipo di dati	Chiave
customer1_subscriptions	subscriptions_index_lvl0	BIGINT	PK
customer1_subscriptions	value	VARCHAR	
customer_address	città	VARCHAR	
customer_address	region	VARCHAR	
customer_address	paese	VARCHAR	
customer_address	code	VARCHAR	

Gestione dei tipi di dati di oggetti e array

Finora, abbiamo solo descritto come vengono mappati i tipi di dati scalari. I tipi di dati Object e Array sono (attualmente) mappati su tabelle virtuali. Il driver JDBC creerà una tabella virtuale per rappresentare i campi oggetto o array in un documento. Il nome della tabella virtuale mappata concatenerà il nome della raccolta originale seguito dal nome del campo separato da un carattere di sottolineatura («_»).

La chiave primaria della tabella base («_id») assume un nuovo nome nella nuova tabella virtuale e viene fornita come chiave esterna alla tabella base associata.

Per i campi di tipo array incorporato, vengono generate colonne di indice per rappresentare l'indice nell'array a ogni livello dell'array.

Esempio di campo oggetto incorporato

Per i campi oggetto in un documento, il driver JDBC crea una mappatura a una tabella virtuale.

```
{
   "Collection: customer",
   "_id":"112233",
   "name":"George Jackson",
   "address":{
      "address1":"123 Avenue Way",
      "address2":"Apt. 5",
```

```
      "city":"Hollywood",
      "region":"California",
      "country":"USA",
      "code":"90210"
   }
}
```

L'esempio precedente è mappato allo schema per la tabella dei clienti:

Nome tabella	Nome colonna	Tipo di dati	Chiave
customer	ID cliente	VARCHAR	PK
customer	nome	VARCHAR	

e la tabella virtuale customer_address:

Nome tabella	Nome colonna	Tipo di dati	Chiave
customer_address	ID cliente	VARCHAR	PK/FK
customer_address	indirizzo 1	VARCHAR	
customer_address	indirizzo 2	VARCHAR	
customer_address	città	VARCHAR	
customer_address	region	VARCHAR	
customer_address	paese	VARCHAR	
customer_address	code	VARCHAR	

Esempio di campo di matrice incorporato

Per i campi di matrice in un documento, il driver JDBC crea anche una mappatura a una tabella virtuale.

```
{
```

```
   "Collection: customer1",
   "_id":"112233",
   "name":"George Jackson",
   "subscriptions":[
      "Vogue",
      "People",
      "USA Today"
   ]
}
```

L'esempio precedente è mappato allo schema della tabella customer1:

Nome tabella	Nome colonna	Tipo di dati	Chiave
cliente1	ID cliente1	VARCHAR	PK
cliente 1	nome	VARCHAR	

e la tabella virtuale customer1_subscriptions:

Nome tabella	Nome colonna	Tipo di dati	Chiave
customer1_subscriptions	ID cliente1	VARCHAR	PK/FK
customer1_subscriptions	subscriptions_index_lvl0	BIGINT	PK
customer1_subscriptions	value	VARCHAR	
customer_address	città	VARCHAR	
customer_address	region	VARCHAR	
customer_address	paese	VARCHAR	
customer_address	code	VARCHAR	

Supporto e limitazioni SQL

Il driver JDBC di Amazon DocumentDB è un driver di sola lettura che supporta un sottoinsieme e alcune estensioni comuni. SQL-92 Per ulteriori informazioni, consulta la documentazione sulle limitazioni SQL e la documentazione sulle limitazioni JDBC.

Risoluzione dei problemi

In caso di problemi con l'utilizzo del driver JDBC di Amazon DocumentDB, consulta la Guida alla risoluzione dei problemi. https://github.com/aws/amazon-documentdb-jdbc-driver/blob/develop/src/markdown/support/troubleshooting-guide.md

Connettiti utilizzando il driver ODBC di Amazon DocumentDB

Il driver ODBC per Amazon DocumentDB fornisce un' SQL-relational interfaccia per gli sviluppatori e consente la connettività da strumenti di BI come Power BI Desktop e Microsoft Excel.

Per informazioni più dettagliate, consulta la documentazione del driver ODBC di Amazon DocumentDB su. GitHub

Argomenti

- Nozioni di base
- Configurazione del driver ODBC di Amazon DocumentDB in Windows
- Connettiti ad Amazon DocumentDB da Microsoft Excel
- Connettiti ad Amazon DocumentDB da Microsoft Power BI Desktop
- Generazione automatica di schemi
- Supporto e limitazioni SQL
- Risoluzione dei problemi

Nozioni di base

Fase 1: Crea cluster Amazon DocumentDB

Se non disponi già di un cluster Amazon DocumentDB, ci sono diversi modi per iniziare.

> **Note**
>
> Amazon DocumentDB è un servizio solo per Virtual Private Cloud (VPC). Se ti connetti da una macchina locale esterna al VPC del cluster, devi creare una connessione SSH a un'istanza Amazon EC2. Per ulteriori informazioni, consulta Connettiti tramite Amazon EC2. Per ulteriori informazioni sul tunneling SSH, consulta Using an SSH Tunnel to Connect to Amazon DocumentDB.

Fase 2. Installazione JRE o JDK

A seconda dell'applicazione BI, potrebbe essere necessario assicurarsi che sul computer sia installata una versione 8 o successiva di JRE o JDK a 64 bit. Puoi scaricare Java SE Runtime Environment 8 qui. https://www.oracle.com/ca-en/java/technologies/downloads/#java8

Fase 3. Scarica il driver ODBC di Amazon DocumentDB

Scarica il driver ODBC di Amazon DocumentDB qui. https://github.com/aws/amazon-documentdb-odbc-driver/releases Scegli il programma di installazione corretto (ad esempio, documentdb-odbc-1.0.0.msi). Segui la guida all'installazione.

Fase 4. Utilizzo di un tunnel SSH per connettersi ad Amazon DocumentDB

I cluster Amazon DocumentDB sono distribuiti all'interno di un Amazon Virtual Private Cloud (Amazon VPC). È possibile accedervi direttamente dalle istanze Amazon EC2 o da altri AWS servizi distribuiti nello stesso Amazon VPC. Inoltre, è possibile accedere ad Amazon DocumentDB da istanze Amazon EC2 o altri AWS servizi in diversi VPC nella stessa o in altre regioni tramite peering VPC. Regione AWS

Tuttavia, supponiamo che il tuo caso d'uso richieda che tu (o la tua applicazione) accedete alle risorse di Amazon DocumentDB dall'esterno del VPC del cluster. Questo accadrà per la maggior parte degli utenti che non eseguono la propria applicazione su una macchina virtuale nello stesso VPC del cluster Amazon DocumentDB. Quando ti connetti dall'esterno del VPC, puoi utilizzare il tunneling SSH (noto anche come port forwarding) per accedere alle tue risorse Amazon DocumentDB.

Per creare un tunnel SSH, è necessaria un'istanza Amazon EC2 in esecuzione nello stesso Amazon VPC del cluster Amazon DocumentDB. Puoi utilizzare un'istanza EC2 esistente nello stesso VPC del cluster o crearne una. Puoi configurare un tunnel SSH verso il cluster Amazon DocumentDB `sample-cluster.node.us-east-1.docdb.amazonaws.com` eseguendo il seguente comando sul tuo computer locale:

```
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
```

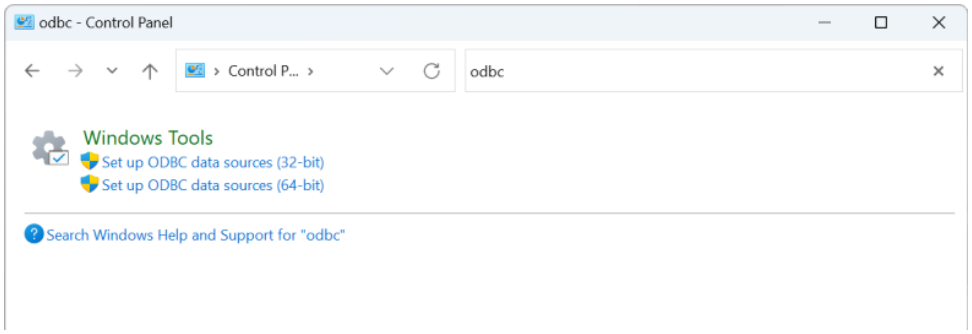
Il flag `-L` viene utilizzato per inoltrare una porta locale. Questo è un prerequisito per la connessione a qualsiasi strumento di BI in esecuzione su un client esterno al tuo VPC. Una volta eseguito il passaggio precedente, puoi passare ai passaggi successivi per lo strumento di BI di tua scelta.

Per ulteriori informazioni sul tunneling SSH, consulta la documentazione sull'uso di un tunnel SSH per connettersi ad Amazon DocumentDB.

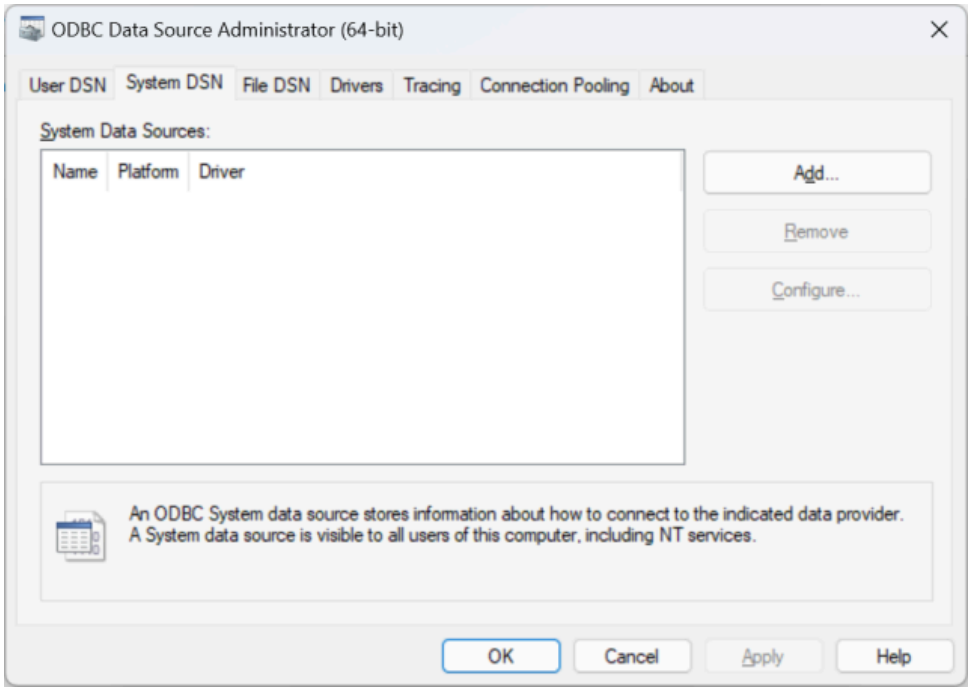
Configurazione del driver ODBC di Amazon DocumentDB in Windows

Utilizza la seguente procedura per configurare il driver ODBC di Amazon DocumentDB in Windows:

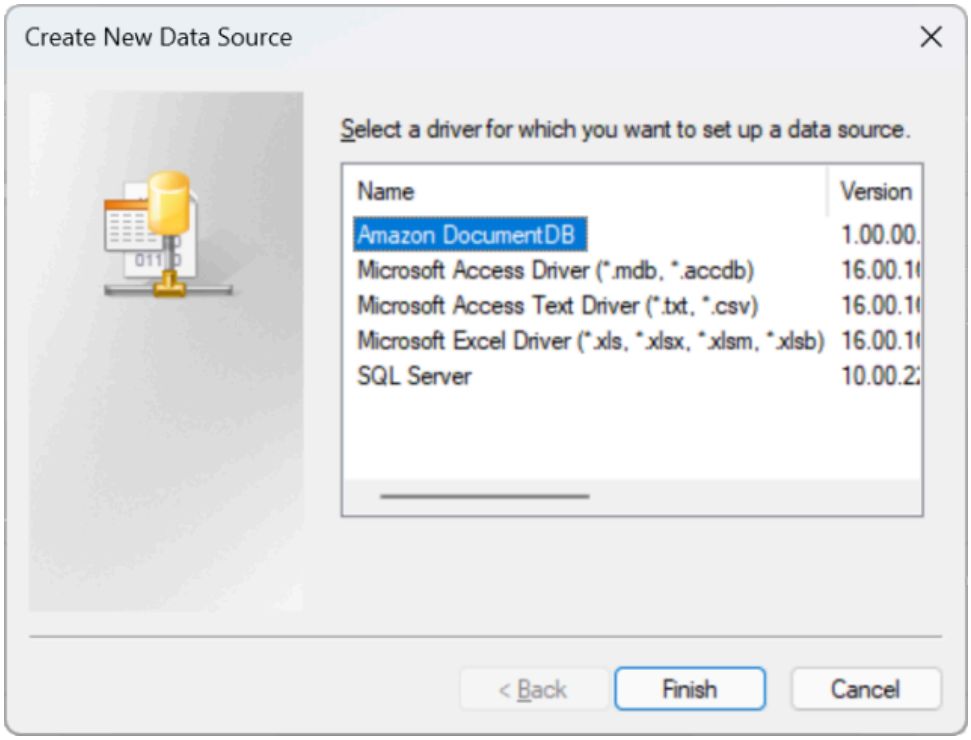
1. Apri il Pannello di controllo in Windows e cerca ODBC (o nel menu, seleziona Strumenti di Windows > Origini dati ODBC (32 bit) o Origini dati ODBC (64 bit)):



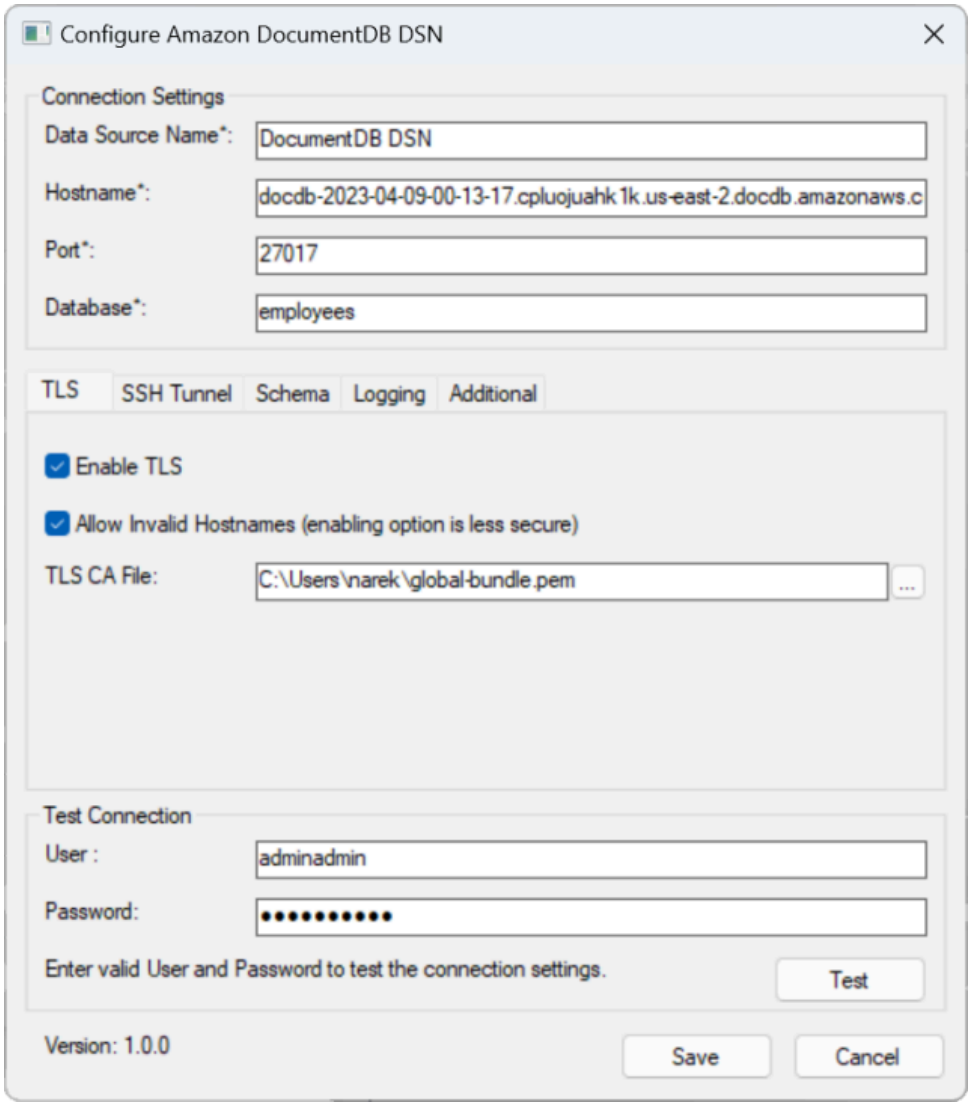
2. Seleziona l'ODBC Driver Data Source Administrator appropriato: opta per la versione a 32 bit se è installata, altrimenti scegli la versione a 64 bit.
3. Seleziona la scheda Sytem DSN e quindi scegli Aggiungi... per aggiungere un nuovo DSN:



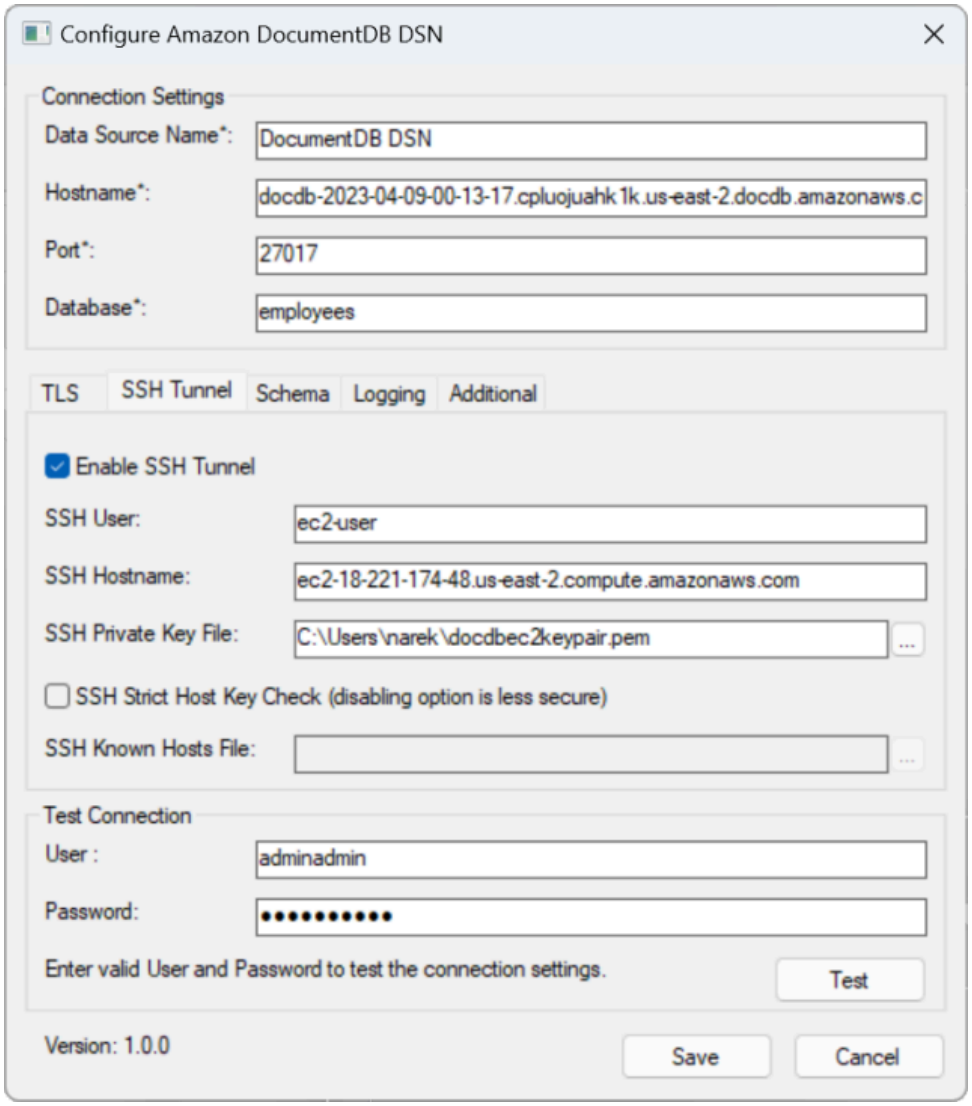
4. Scegli Amazon DocumentDB dall'elenco dei driver delle sorgenti dati:



5. Nella finestra di dialogo Configura Amazon DocumentDB DSN, completa i campi Impostazioni di connessione, scheda TLS e Test Connection, quindi scegli Salva:



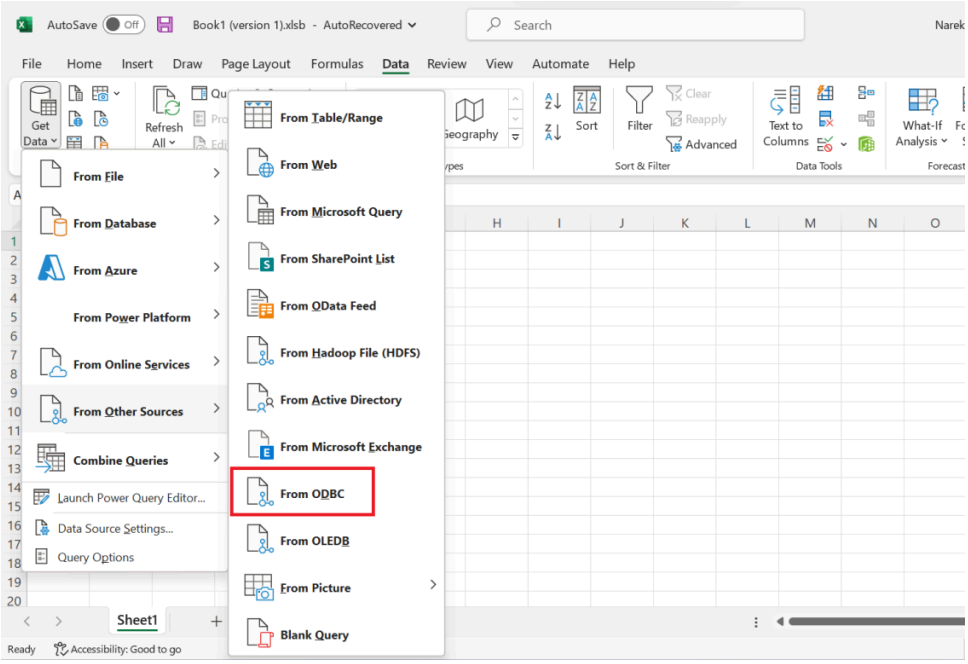
6. Assicurati di completare il modulo Windows in modo accurato, poiché i dettagli della connessione differiranno a seconda del metodo di tunneling SSH scelto verso l'istanza EC2. Vedi i metodi di tunneling SSH qui. https://github.com/aws/amazon-documentdb-odbc-driver/blob/develop/src/markdown/setup/setup.md#using-an-ssh-tunnel-to-connect-to-amazon-documentdb Vedi Sintassi e opzioni delle stringhe di connessione per ulteriori informazioni su ciascuna proprietà.



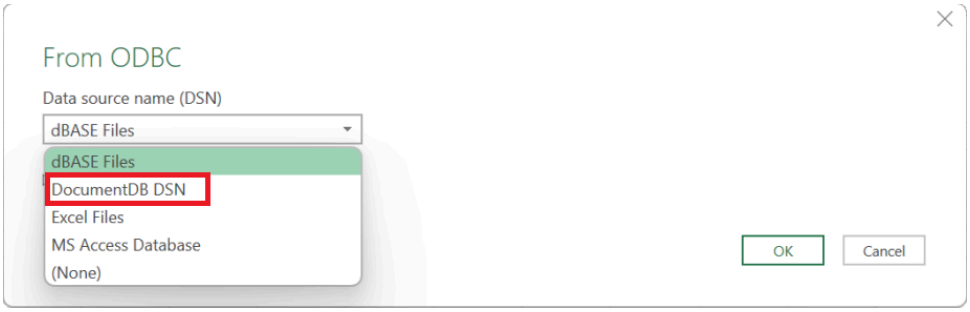
Per ulteriori informazioni sulla configurazione del driver ODBC di Amazon DocumentDB su Windows, consulta questa pagina. https://github.com/aws/amazon-documentdb-odbc-driver/blob/develop/src/markdown/setup/windows-dsn-configuration.md

Connettiti ad Amazon DocumentDB da Microsoft Excel

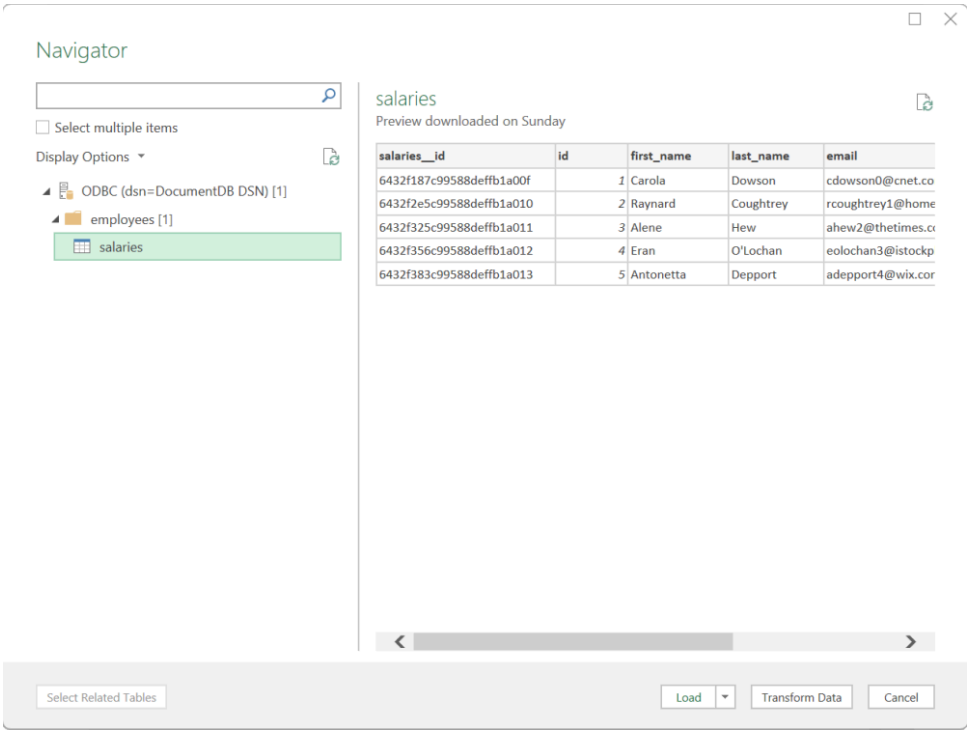
1. Assicurati che il driver Amazon DocumentDB sia stato installato e configurato correttamente. Per ulteriori informazioni, consulta Configurazione del driver ODBC di Amazon DocumentDB in Windows.
2. Avvia Microsoft Excel.
3. Vai a Dati > Ottieni dati > Da altre fonti.
4. Scegli Da ODBC:



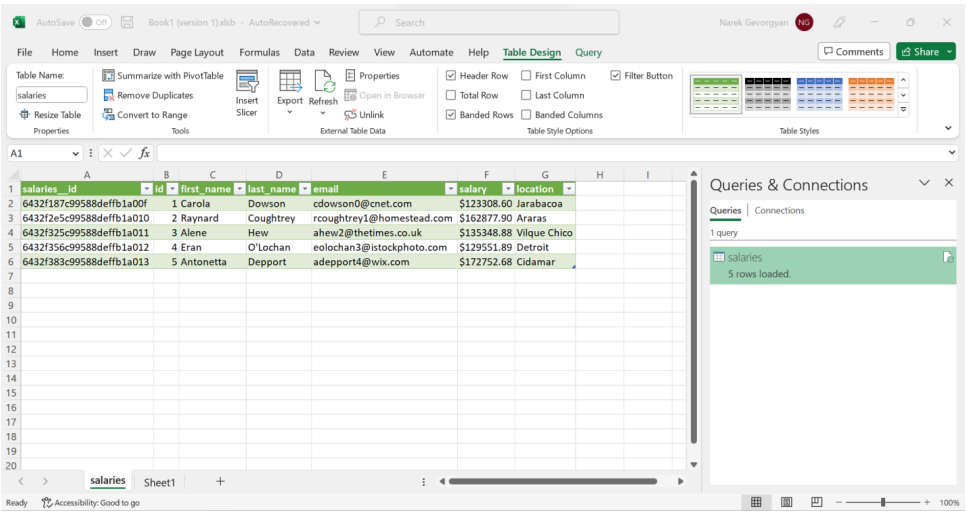
5. Seleziona l'origine dati dal menu a discesa Nome origine dati (DSN) associato ad Amazon DocumentDB:



6. Scegli la raccolta da cui vuoi caricare i dati in Excel:



7. Carica i dati in Excel:



Connettiti ad Amazon DocumentDB da Microsoft Power BI Desktop

Argomenti

- Prerequisiti
- Aggiungere un connettore personalizzato Microsoft Power BI Desktop
- Connessione tramite il connettore personalizzato Amazon DocumentDB
- Configurazione di Microsoft Power BI Gateway

Prerequisiti

Prima di iniziare, assicurati che il driver ODBC di Amazon DocumentDB sia installato correttamente.

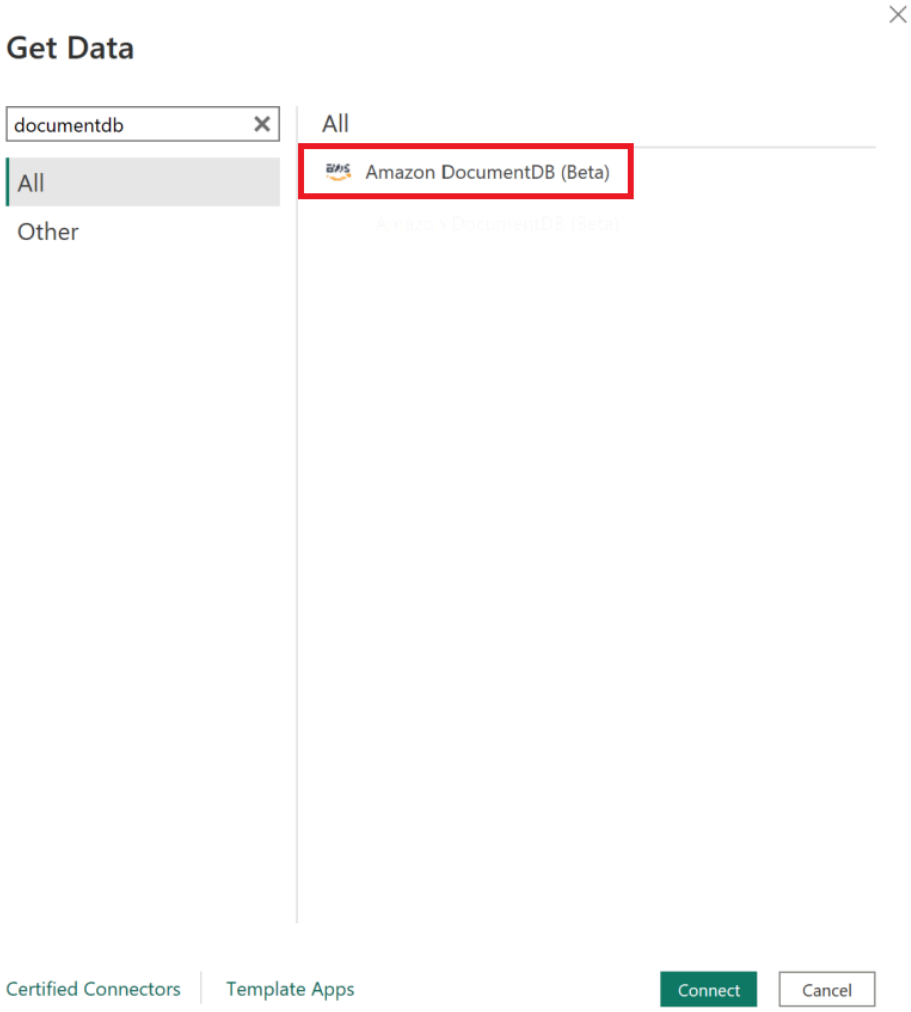
Aggiungere un connettore personalizzato Microsoft Power BI Desktop

Copia il `AmazonDocumentDBConnector.mez` file nella `<User>\Documents\Power BI Desktop\Custom Connectors\` cartella (o nella cartella in `<User>\OneDrive\Documents\Power BI Desktop\Custom Connectors` uso OneDrive). Ciò consentirà a Power BI di accedere al connettore personalizzato. Puoi ottenere il connettore per Power BI Desktop qui. Riavvia Power BI Desktop per assicurarti che il connettore sia caricato.

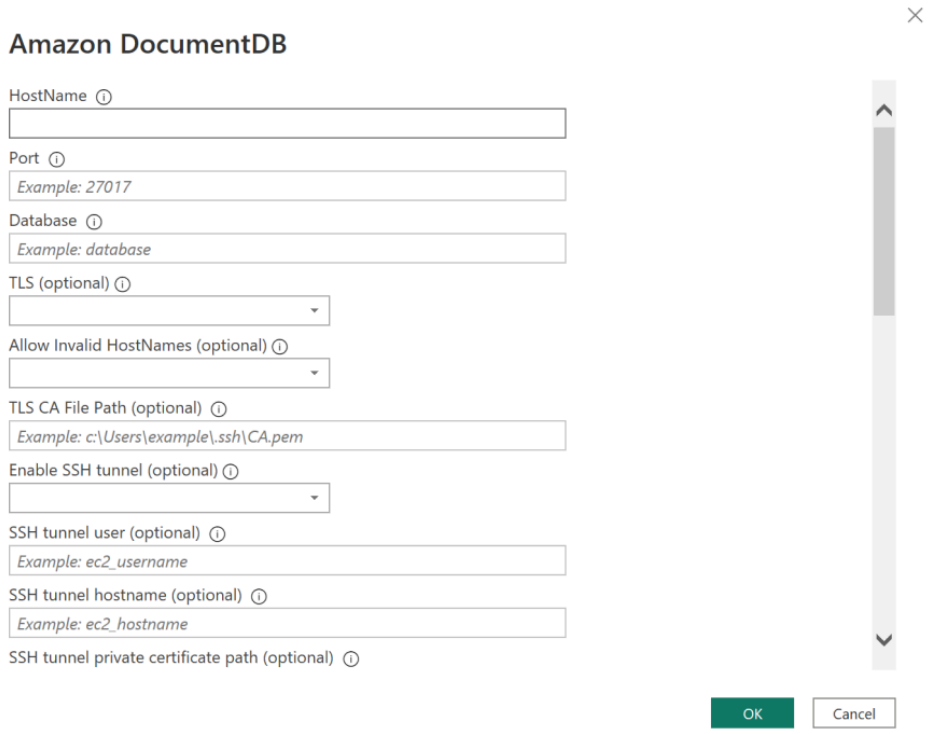
> **Note**
>
> Il connettore personalizzato supporta solo il nome utente e la password di Amazon DocumentDB per l'autenticazione.

Connessione tramite il connettore personalizzato Amazon DocumentDB

1. Seleziona Amazon DocumentDB (Beta) da Get Data e scegli Connetti. Se ricevi un avviso relativo all'utilizzo di un servizio di terze parti, scegli Continua.



2. Inserisci tutte le informazioni necessarie per connetterti al tuo cluster Amazon DocumentDB, quindi scegli OK:



> **Note**
>
> A seconda della configurazione del Data Source Name (DSN) del driver ODBC, la schermata dei dettagli della connessione SSH potrebbe non essere visualizzata se hai già fornito le informazioni necessarie nelle impostazioni DSN.

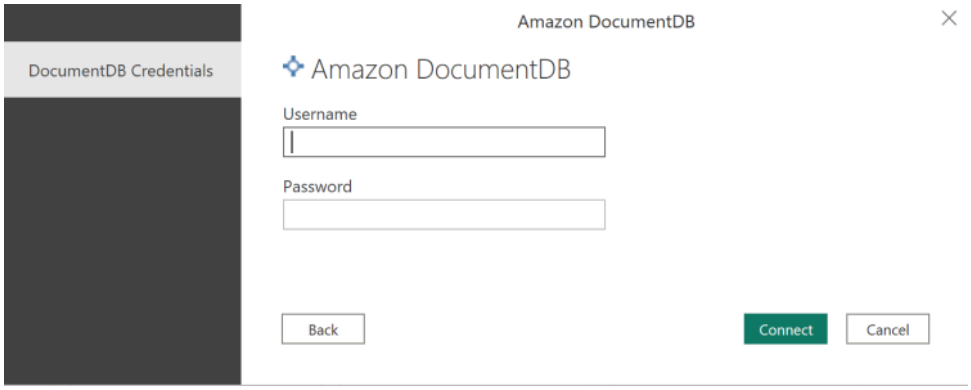
3. Scegli la modalità di connettività dati:

   - Importazione: carica tutti i dati e memorizza le informazioni su disco. I dati devono essere aggiornati e ricaricati per mostrare gli aggiornamenti dei dati.
   - Direct Query: non carica i dati, ma esegue interrogazioni in tempo reale sui dati. Ciò significa che non è necessario aggiornare e ricaricare i dati per mostrare gli aggiornamenti dei dati.

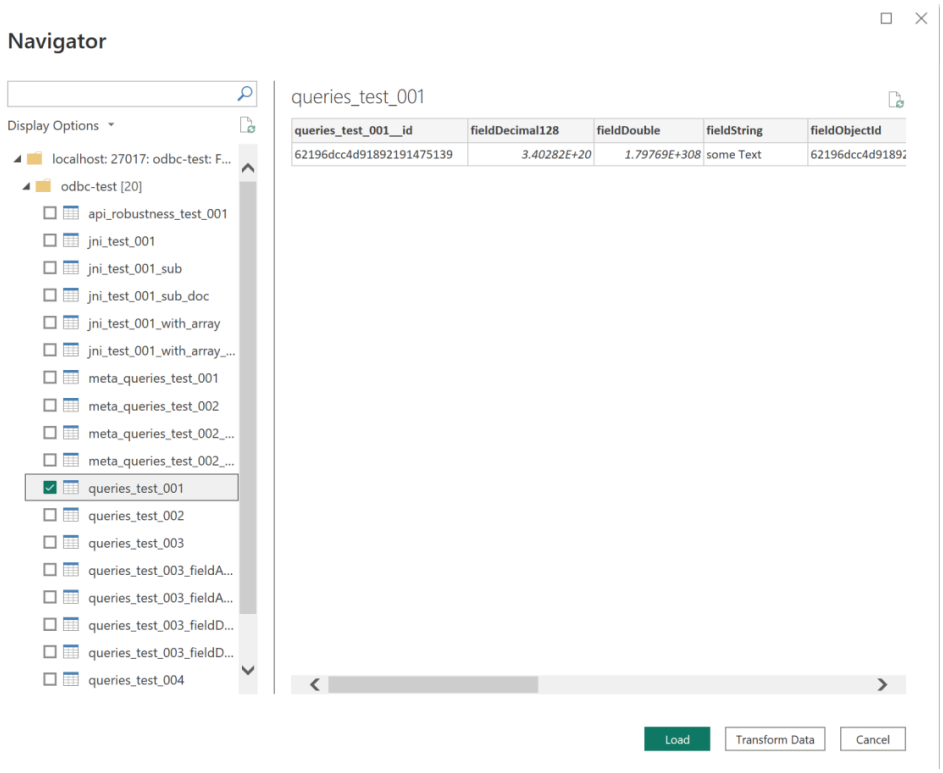


> **Note**
>
> Se si utilizza un set di dati molto grande, l'importazione di tutti i dati potrebbe richiedere un periodo di tempo più lungo.

4. Se è la prima volta che ti connetti a questa origine dati, seleziona il tipo di autenticazione e inserisci le tue credenziali quando richiesto. Quindi scegli Connetti:



5. Nella finestra di dialogo Navigator, selezionate le tabelle del database desiderate, quindi scegliete Carica per caricare i dati o Trasforma dati per continuare a trasformare i dati.



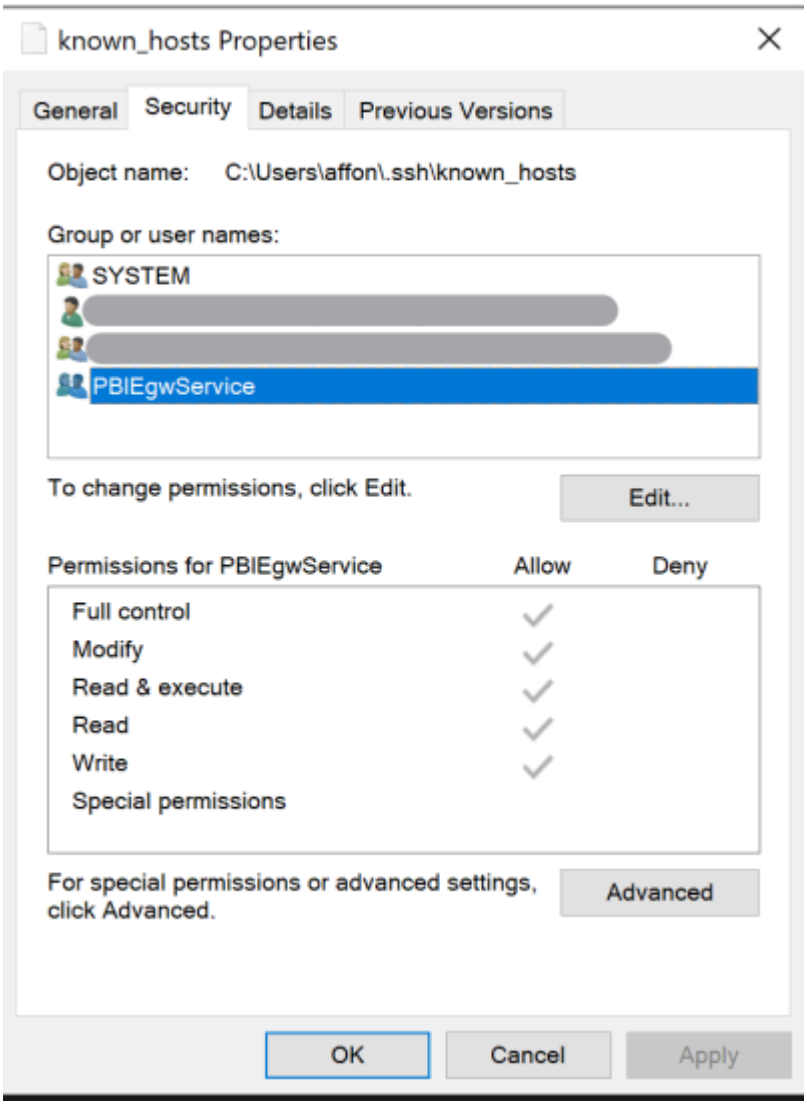
> **Note**
>
> Le impostazioni dell'origine dati vengono salvate una volta effettuata la connessione. Per modificarle, seleziona Trasforma dati > Impostazioni origine dati.

Configurazione di Microsoft Power BI Gateway

Prerequisiti:

- Assicurati che il connettore personalizzato funzioni con Power BI Gateway.
- Assicurati che il DSN ODBC sia creato nelle origini dati ODBC nella scheda Sistema sul computer in cui è installato Power BI Gateway.

Se si utilizza la funzionalità di tunnel SSH interno, il file `known_hosts` deve trovarsi dove l'account del servizio Power BI può accedervi.



> **Note**
>
> Ciò vale anche per qualsiasi file di cui potresti aver bisogno per stabilire una connessione al tuo cluster Amazon DocumentDB, ad esempio un file di certificato dell'autorità di certificazione (CA) (file pem).

Generazione automatica di schemi

Il driver ODBC utilizza il driver JDBC di Amazon DocumentDB tramite JNI (Java Native Interface), facendo sì che la funzionalità di generazione automatica dello schema funzioni in modo simile nel driver JDBC. Per ulteriori informazioni sulla generazione automatica di schemi, vedereGenerazione

automatica di schemi JDBC. Per ulteriori informazioni sull'architettura dei driver ODBC, consulta l'architettura dei driver ODBC di Amazon DocumentDB su. GitHub

Supporto e limitazioni SQL

Il driver ODBC di Amazon DocumentDB è un driver di sola lettura che supporta un sottoinsieme e alcune estensioni comuni. SQL-92 Per ulteriori informazioni, consulta la documentazione relativa al supporto e alle limitazioni ODBC.

Risoluzione dei problemi

In caso di problemi con l'utilizzo del driver ODBC di Amazon DocumentDB, consulta la Guida alla risoluzione dei problemi.

Programma con Amazon DocumentDB

Il servizio supporta la convalida dello schema JSON.

Argomenti

- Guida alla programmazione Java di Amazon DocumentDB
- Utilizzo della convalida dello schema JSON

Guida alla programmazione Java di Amazon DocumentDB

Questa guida completa fornisce una guida dettagliata per lavorare con Amazon DocumentDB utilizzando i driver Java di MongoDB, coprendo aspetti essenziali delle operazioni e della gestione del database.

Argomenti

- Introduzione
- Prerequisiti
- Modelli di dati per questa guida
- Connessione ad Amazon DocumentDB con un driver Java MongoDB
- Esecuzione di operazioni CRUD in Amazon DocumentDB con Java
- Gestione degli indici in Amazon DocumentDB con Java
- Event-driven programmazione con Amazon DocumentDB e Java

Introduzione

La guida inizia con la connettività e spiega come stabilire connessioni sicure ai cluster DocumentDB utilizzando il driver Java MongoDB. Descrive in dettaglio i componenti della stringa di connessione, SSL/TLS l'implementazione e le varie opzioni di connessione, tra cui l'autenticazione IAM e il pool di connessioni, insieme a solide strategie di gestione degli errori.

Nella sezione delle operazioni CRUD (create, read, update, delete), la guida illustra in modo completo la manipolazione dei documenti, dimostrando come creare, leggere, aggiornare ed eliminare documenti utilizzando operazioni singole o in blocco. Spiega l'uso di filtri, interrogazioni e varie opzioni operative, enfatizzando al contempo le migliori pratiche per la gestione degli errori e l'implementazione della logica dei tentativi per una maggiore affidabilità. La guida illustra inoltre ampiamente la gestione degli indici, descrivendo in dettaglio la creazione e la manutenzione di diversi tipi di indice, tra cui indici a campo singolo, composti, sparsi e di testo. Spiega come ottimizzare le prestazioni delle query attraverso una corretta selezione degli indici e l'utilizzo della `explain()` funzione per analizzare i piani di esecuzione delle query.

La sezione finale si concentra sulla programmazione basata sugli eventi utilizzando i flussi di modifica di Amazon DocumentDB, dimostrando come implementare il monitoraggio delle modifiche dei dati in tempo reale nelle applicazioni Java. Copre l'implementazione dei cursori del flusso di modifica, la gestione dei token di curriculum per un funzionamento continuo e le operazioni basate sul tempo per l'elaborazione dei dati storici. In tutta la guida, vengono forniti esempi pratici di codice e best practice, che la rendono una risorsa inestimabile per la creazione di solide applicazioni Java con Amazon DocumentDB.

Prerequisiti

Prima di iniziare, assicurati di disporre di:

- E Account AWS con un cluster Amazon DocumentDB configurato. Per la configurazione del cluster, consulta Getting started with Amazon DocumentDB using Amazon EC2 nel AWS Database Blog.
- Java Development Kit (JDK) installato (utilizzeremo Amazon Corretto 21 per questa guida).
- Maven per la gestione delle dipendenze.

Modelli di dati per questa guida

Tutto il codice di esempio in questa guida presuppone una connessione a un database di test «ProgGuideData» che contiene una raccolta «Restaurants». Tutti i codici di esempio in questa guida funzionano su un sistema di elenchi di ristoranti e di seguito è riportato un esempio di come appare un documento in questo sistema:

```
{
    "_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
    }
}
```

Tutti gli esempi di codice che mostrano CRUD, la gestione degli indici e la programmazione basata sugli eventi presuppongono la presenza di un `MongoClient` oggettodb`Client`, un oggetto e un `MongoDatabase` oggetto`connection`DB. `MongoCollection` `collection`

> **Note**
>
> Tutti gli esempi di codice in questa guida sono stati testati con il driver Java MongoDB versione 5.3.0.

Connessione ad Amazon DocumentDB con un driver Java MongoDB

Questa sezione fornisce una guida dettagliata per la connessione ad Amazon DocumentDB utilizzando i driver Java. Questo ti consentirà di iniziare a integrare DocumentDB nelle tue applicazioni Java.

Argomenti

- Fase 1: Configurazione del progetto
- Passaggio 2: creare la stringa di connessione
- Fase 3: Scrivere il codice di connessione
- Fase 4: Gestire le eccezioni di connessione
- Fase 5: Esecuzione del codice
- Le migliori pratiche di connessione

Fase 1: Configurazione del progetto

1. Usando Maven, crea un progetto java:

   ```
   mvn archetype:generate -DgroupId=com.docdb.guide -DartifactId=my-docdb-project -
   DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
   ```

2. Aggiungi il driver Java MongoDB come dipendenza per il progetto nel tuo file 'pom.xml':

   ```
   <dependency>
   ```

```
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>5.3.0</version>
</dependency>
```

Passaggio 2: creare la stringa di connessione

La stringa di connessione Amazon DocumentDB è essenziale per stabilire una connessione tra l'applicazione e il cluster DocumentDB. Questa stringa racchiude informazioni cruciali come l'endpoint del cluster, la porta, i dettagli di autenticazione e varie opzioni di connessione. Per creare una stringa di connessione DocumentDB, in genere si inizia con il formato di base:

```
"mongodb://username:password@cluster-endpoint:port/?[connection options]"
```

Dovrai sostituire «nome utente» e «password» con le tue credenziali effettive. Puoi trovare l'endpoint e il numero di porta del tuo cluster Console di gestione AWS in e tramite. AWS CLI Consulta Individuazione degli endpoint di un cluster per trovare l'endpoint del cluster per il tuo cluster. La porta predefinita per DocumentDB è 27017.

Esempi di stringhe di connessione

- Effettuare una connessione a DocumentDB utilizzando la crittografia in transito e assicurandosi che le richieste di lettura vadano alle repliche di lettura e scrivano al primario:

```
"mongodb://username:password@cluster-endpoint:27017/?tls=true&
   tlsCAFile=global-bundle.pem&
   readPreference=secondaryPreferred&
   retryWrites=false"
```

- Effettuare una connessione a DocumentDB utilizzando l'autenticazione IAM:

```
"mongodb://cluster-endpoint:27017/?tls=true&
   tlsCAFile=global-bundle.pem&
   readPreference=secondaryPreferred&
   retryWrites=false&
   authSource=%24external&
   authMechanism=MONGODB-AWS"
```

Le diverse opzioni disponibili per la stringa di connessione sono le seguenti:

- Certificato TLS
- Lettura da repliche lette
- Scrivi preoccupazioni e scrivi nel diario
- RetryWrites
- Autenticazione IAM
- Pool di connessioni
- Parametri di timeout della connessione

Certificato TLS

`tls=true|false`— Questa opzione abilita o disabilita Transport Layer Security (TLS). Per impostazione predefinita, la crittografia in transito è abilitata sul cluster Amazon DocumentDB e pertanto, a meno che TLS non sia disabilitato a livello di cluster, il valore di questa opzione dovrebbe essere. `true`

Quando si utilizza TLS, il codice deve fornire un certificato SSL durante la creazione della connessione a un cluster DocumentDB. Scarica il certificato necessario per effettuare la connessione sicura al cluster:. `global-bundle.pem` Esistono due modi per utilizzare il `global-bundle.pem` file.

- Opzione 1: estrarre tutti i certificati dal `global-bundle.pem` file e utilizzare il keytool di Java per archiviarli in un `.jks` file che può essere successivamente utilizzato nel codice. Fai riferimento alla scheda Java Connessione con TLS abilitato per lo script che mostra come eseguire questa operazione.
- Opzione 2: aggiungete dinamicamente il `global-bundle.pem` file nel codice, create un keystore in memoria e utilizzatelo `SSLContext` per fornire il certificato come parte della connessione.

Lettura da repliche lette

`replicaSet=rs0&readPreference=secondaryPreferred`— La specificazione di queste due opzioni indirizza tutte le richieste di lettura alle repliche di lettura e le richieste di scrittura all'istanza primaria. L'utilizzo `replicaSet=rs0` della stringa di connessione consente al driver MongoDB di mantenere una vista aggiornata automaticamente della topologia del cluster, consentendo alle applicazioni di mantenere la visibilità delle configurazioni correnti dei nodi man mano che le istanze vengono aggiunte o rimosse. La mancata fornitura di queste opzioni o la specifica

`readPreference=primary` invia tutte le letture e le scritture all'istanza primaria. Per ulteriori opzioni per`readPreference`, vedereLeggi le opzioni di preferenza.

Scrivi preoccupazioni e scrivi nel diario

Il problema di scrittura determina il livello di riconoscimento richiesto dal database per le operazioni di scrittura. I driver MongoDB forniscono un'opzione per ottimizzare i problemi di scrittura e i file di diario. Amazon DocumentDB non si aspetta che tu imposti il criterio di scrittura e il journal e ignora i valori inviati per `w` e (`ej`). `writeConcern` `journal` DocumentDB scrive sempre i dati con`writeConcern`: `majority` `ejournal`: `true` in modo che le scritture vengano registrate in modo duraturo sulla maggior parte dei nodi prima di inviare una conferma al client.

RetryWrites

`retryWrites`— A partire dalla versione 8.0.2 del motore, Amazon DocumentDB supporta le scritture riutilizzabili. Impostato per abilitare. `retryWrites=true` Nelle versioni del motore precedenti alla 8.0.2, imposta questo attributo `false` su. Per ulteriori informazioni, consulta Retryable scrive in Amazon DocumentDB.

Autenticazione IAM

`authSource=%24external`e **`authMechanism=MONGODB-AWS`** — Questi due parametri vengono utilizzati per l'autenticazione utilizzando. AWS Identity and Access Management L'autenticazione IAM è attualmente disponibile solo nella versione 5.0 del cluster basato su istanze. Per ulteriori informazioni, consulta Autenticazione tramite identità IAM.

Pool di connessioni

Queste opzioni sono disponibili per il pool di connessioni:

- **`maxPoolSize`**— Imposta il numero massimo di connessioni che possono essere create nel pool. Quando tutte le connessioni sono in uso e arriva una nuova richiesta, attende che una connessione diventi disponibile. L'impostazione predefinita per i driver Java MongoDB è 100.
- **`minPoolSize`**— Indica il numero minimo di connessioni che devono essere mantenute nel pool in ogni momento. L'impostazione predefinita per i driver Java MongoDB è 0.
- **`maxIdleTimeMS`**— Determina per quanto tempo una connessione può rimanere inattiva nel pool prima di essere chiusa e rimossa. L'impostazione predefinita per i driver Java MongoDB è 100 millisecondi.
- **`waitQueueTimeoutMS`**— Configura per quanto tempo un thread deve attendere che una connessione diventi disponibile quando il pool è alla sua dimensione massima. Se una

connessione non diventa disponibile entro questo periodo, viene generata un'eccezione. Il valore predefinito per i driver Java MongoDB è 120.000 millisecondi (2 minuti).

Parametri di timeout della connessione

Il timeout è un meccanismo che limita il tempo necessario per un'operazione o un tentativo di connessione prima che venga considerato fallito. Sono disponibili i seguenti parametri di timeout per prevenire attese indefinite e gestire l'allocazione delle risorse:

- **`connectTimeoutMS`**— Configura il tempo di attesa del driver per stabilire una connessione al cluster. L'impostazione predefinita è 10.000 millisecondi (10 secondi).
- **`socketTimeoutMS`**— Specifica per quanto tempo il driver attenderà una risposta dal server per un'operazione non di scrittura. L'impostazione predefinita è 0 (nessun timeout o infinito).
- **`serverSelectionTimeoutMS`**— Specifica il tempo di attesa del driver per trovare un server disponibile nel cluster. Il valore predefinito per questa impostazione è 30 secondi ed è sufficiente per l'elezione di una nuova istanza primaria durante il failover.

Fase 3: Scrivere il codice di connessione

Il seguente esempio di codice mostra come creare una connessione TLS ad Amazon DocumentDB:

- Crea oggetti Java `KeyStore` e `SSLContext >`.
- Inoltre crea l'`MongoClientSettings`oggetto passandolo all'`ConnectionString`oggetto. Per stabilire una connessione TLS, è necessario utilizzare l'`MongoClientSettings`oggetto per associare e. `connectionstring` `sslcontext`
- Usando `MongoClients` si ottiene un oggetto `MongoClient`.

```
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);
    }
}
```

Fase 4: Gestire le eccezioni di connessione

Quando si lavora con DocumentDB in applicazioni Java, la gestione delle eccezioni di connessione è fondamentale per mantenere operazioni di database solide e affidabili. Se gestite correttamente, queste eccezioni non solo aiutano a diagnosticare rapidamente i problemi, ma assicurano anche che l'applicazione possa gestire con garbo le interruzioni temporanee della rete o l'indisponibilità del server, migliorando la stabilità e l'esperienza utente. Alcune delle eccezioni critiche relative alla creazione della connessione includono:

- **`MongoException`**— Un'eccezione generale e potrebbe essere emessa in vari scenari non coperti da eccezioni più specifiche. Assicurati che questa eccezione venga gestita dopo tutte le altre eccezioni specifiche poiché si tratta di un'eccezione generale di Catch all MongoDB.
- **`MongoTimeoutException`**— Emesso quando un'operazione scade. Ad esempio, interrogare un endpoint del cluster inesistente.

- **`MongoSocketException`**— Rilasciato per problemi relativi alla rete. Ad esempio, un'improvvisa disconnessione dalla rete durante un'operazione.
- **`MongoSecurityException`**— Emesso quando l'autenticazione fallisce. Ad esempio, connessione con credenziali errate.
- **`MongoConfigurationException`**— Emesso in caso di errore nella configurazione del client. Ad esempio, utilizzando una stringa di connessione non valida.

Fase 5: Esecuzione del codice

Il seguente esempio di codice crea una connessione Amazon DocumentDB e stampa tutti i database:

```
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);
    }
}
```

Le migliori pratiche di connessione

Le seguenti sono le best practice da prendere in considerazione quando ci si connette ad Amazon DocumentDB con un driver Java MongoDB:

- Chiudi sempre il tuo `MongoClient` quando non hai più bisogno del client per rilasciare risorse.
- Gestisci le eccezioni in modo appropriato e implementa una corretta registrazione degli errori.
- Utilizza le variabili di ambiente o AWS Secrets Manager per archiviare informazioni sensibili come nomi utente e password.

Esecuzione di operazioni CRUD in Amazon DocumentDB con Java

Questa sezione illustra l'esecuzione delle operazioni CRUD (creazione, lettura, aggiornamento, eliminazione) in Amazon DocumentDB utilizzando i driver Java MongoDB.

Argomenti

- Creazione e inserimento di documenti in una raccolta DocumentDB
- Lettura e recupero di dati da una raccolta DocumentDB
- Aggiornamento dei documenti esistenti in una raccolta DocumentDB
- Rimozione di documenti da una raccolta DocumentDB
- Gestione degli errori con logica di riprova

Creazione e inserimento di documenti in una raccolta DocumentDB

L'inserimento di documenti in Amazon DocumentDB ti consente di aggiungere nuovi dati alle tue raccolte. Esistono diversi modi per eseguire gli inserimenti, a seconda delle esigenze e del volume di dati con cui stai lavorando. Il metodo più semplice per inserire un singolo documento nella raccolta è. `insertOne()` Per inserire più documenti contemporaneamente, è possibile utilizzare il `insertMany()` metodo, che consente di aggiungere una serie di documenti in un'unica operazione. Un altro metodo per inserire molti documenti in una raccolta DocumentDB è. `bulkWrite()` In questa guida, discutiamo tutti questi metodi per creare documenti in una raccolta DocumentDB.

`insertOne()`

Cominciamo esaminando come inserire un singolo documento in una raccolta Amazon DocumentDBB. L'inserimento di un singolo documento viene eseguito utilizzando il metodo. `insertOne()` Questo metodo richiede un metodo BsonDocument di inserimento e restituisce un `InsertOneResult` oggetto che può essere utilizzato per ottenere l'ID oggetto del nuovo documento inserito. Il codice di esempio seguente mostra l'inserimento di un documento relativo a un ristorante nella raccolta:

```
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;
}
```

Durante l'utilizzo`insertOne()`, assicuratevi di includere una gestione degli errori appropriata. Ad esempio, nel codice precedente, «`restaurantId`» ha un indice univoco e quindi eseguendo nuovamente questo codice si otterrà quanto segue`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={}}.
```

insertMany ()

I metodi principali utilizzati per inserire molti documenti in una raccolta sono insertMany () e. `bulkWrite()`

Il `insertMany()` metodo è il modo più semplice per inserire più documenti in un'unica operazione. Accetta un elenco di documenti e li inserisce nella raccolta. Questo metodo è ideale quando si inserisce una serie di nuovi documenti indipendenti l'uno dall'altro e che non richiedono alcuna elaborazione speciale o operazioni miste. Il codice seguente mostra la lettura di documenti JSON da un file e il loro inserimento nella raccolta. La `insertMany()` funzione restituisce un [InsertManyResult](#) `InsertManyResult` oggetto che può essere utilizzato per ottenere gli ID di tutti i documenti inseriti.

```
// 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()

Il `bulkWrite()` metodo consente di eseguire più operazioni di scrittura (inserimento, aggiornamento, eliminazione) in un unico batch. È possibile utilizzarlo `bulkWrite()` quando è necessario eseguire diversi tipi di operazioni in un singolo batch, ad esempio inserire alcuni documenti e aggiornarne altri. `bulkWrite()`supporta due tipi di scrittura in batch, ordinata e non ordinata:

- Operazioni ordinate: (impostazione predefinita) Amazon DocumentDB elabora le operazioni di scrittura in sequenza e si interrompe al primo errore riscontrato. Ciò è utile quando l'ordine delle operazioni è importante, ad esempio quando le operazioni successive dipendono da quelle precedenti. Tuttavia, le operazioni ordinate sono generalmente più lente delle operazioni non ordinate. Con le operazioni ordinate, è necessario risolvere il caso in cui il batch si interrompa al primo errore, lasciando potenzialmente alcune operazioni non elaborate.
- Operazioni non ordinate: consente ad Amazon DocumentDB di elaborare gli inserimenti come un'unica esecuzione nel database. Se si verifica un errore con un documento, l'operazione continua con i documenti rimanenti. Ciò è particolarmente utile quando si inseriscono grandi quantità di dati e si possono tollerare alcuni errori, ad esempio durante la migrazione dei dati o le importazioni di massa in cui alcuni documenti potrebbero non riuscire a causa di chiavi duplicate. Nel caso di operazioni non ordinate, è necessario affrontare scenari di successo parziale in cui alcune operazioni hanno esito positivo mentre altre falliscono.

Quando si lavora con il `bulkWrite()` metodo, sono necessarie alcune classi essenziali. Innanzitutto, la `WriteModel` classe funge da classe base per tutte le operazioni di scrittura e con implementazioni specifiche come `InsertOneModel`, `UpdateOneModel` `UpdateManyModel` `DeleteOneModel`, e per la `DeleteManyModel` gestione di diversi tipi di operazioni.

La `BulkWriteOptions` classe è necessaria per configurare il comportamento delle operazioni in blocco, ad esempio impostare ordered/unordered l'esecuzione o bypassare la convalida dei documenti. La `BulkWriteResult` classe fornisce informazioni dettagliate sui risultati dell'esecuzione, incluso il numero di documenti inseriti, aggiornati ed eliminati.

Per la gestione degli errori, la `MongoBulkWriteException` classe è fondamentale in quanto contiene informazioni su eventuali errori durante l'operazione in blocco, mentre la `BulkWriteError` classe fornisce dettagli specifici sugli errori delle singole operazioni. Il codice seguente mostra un esempio di inserimento di un elenco di documenti, nonché di aggiornamento ed eliminazione di un singolo documento, il tutto nell'esecuzione di una singola chiamata al metodo. `bulkWrite()` Il codice mostra anche come utilizzare `BulkWriteOptions` e `BulkWriteResult` gestire correttamente gli errori dell'`bulkWrite()`operazione.

```
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());
}
```

Retryable scrive

A differenza di MongoDB, Amazon DocumentDB non supporta scritture riutilizzabili. Di conseguenza, è necessario implementare una logica di riprova personalizzata nelle loro applicazioni, in particolare per gestire problemi di rete o indisponibilità temporanea del servizio. Una strategia di ripetizione ben implementata, in genere, comporta l'aumento del ritardo tra i tentativi e la limitazione del numero totale di tentativi. Gestione degli errori con logica di riprovaDi seguito è riportato un esempio di codice per la creazione di una logica di ripetizione con gestione degli errori.

Lettura e recupero di dati da una raccolta DocumentDB

L'interrogazione dei documenti in Amazon DocumentDB si basa su diversi componenti chiave che consentono di recuperare e manipolare i dati con precisione. Il `find()` metodo è l'API di interrogazione fondamentale nei driver Java di MongoDB. Consente il recupero di dati complessi con numerose opzioni per filtrare, ordinare e proiettare i risultati. Oltre al `find()` metodo, `Filters` `FindIterable` ci sono altri due componenti fondamentali che forniscono gli elementi costitutivi per le operazioni di interrogazione nei driver Java MongoDB.

La `Filters` classe è una classe di utilità nel driver Java MongoDB che fornisce un'API fluente per la creazione di filtri di interrogazione. Questa classe offre metodi di fabbrica statici che creano istanze di `Bson` oggetti che rappresentano varie condizioni di interrogazione. I metodi più comunemente usati includono `eq()` per i confronti di uguaglianza, `gt()` `lt()``gte()`, e `lte()` per i confronti numerici e per la combinazione di più condizioni, `and()` `or()` per i test di appartenenza agli array `in()` e `nin()` per la corrispondenza dei modelli. `regex()` La classe è progettata per resistere ai tipi e fornisce un migliore controllo in fase di compilazione rispetto alle query non elaborate basate su documenti, rendendola l'approccio preferito per la creazione di query DocumentDB nelle applicazioni Java. La gestione degli errori è robusta, con chiare eccezioni generate per le costruzioni di filtri non valide.

`FindIterable`è un'interfaccia specializzata progettata per gestire il risultato del metodo. `find()` Fornisce un ricco set di metodi per perfezionare e controllare l'esecuzione delle query, offrendo un'API fluida per il concatenamento dei metodi. L'interfaccia include metodi essenziali di modifica delle query, ad esempio `limit()` per limitare il numero di documenti restituiti, per l'impaginazione, `skip()` per l'ordinamento dei risultati, `sort()` per la selezione di campi specifici e `projection()` per la selezione dell'indice. `hint()` Le operazioni batch, skip e limit in `FindIterable` sono strumenti essenziali di impaginazione e gestione dei dati che aiutano a controllare il modo in cui i documenti vengono recuperati ed elaborati dal database.

Batching (`batchSize`) controlla quanti documenti DocumentDB restituisce al client in un singolo round trip di rete. Quando si imposta una dimensione del batch, DocumentDB non restituisce tutti i documenti corrispondenti contemporaneamente, ma li restituisce in gruppi della dimensione batch specificata.

Skip consente di compensare il punto di partenza dei risultati, dicendo essenzialmente a DocumentDB di saltare un numero specificato di documenti prima di iniziare a restituire le corrispondenze. Ad esempio, `skip(20)` ignorerà i primi 20 documenti corrispondenti. Questo è comunemente usato negli scenari di impaginazione in cui si desidera recuperare pagine successive di risultati.

Limite limita il numero totale di documenti che possono essere restituiti da una query. Quando lo specifichi`limit(n)`, DocumentDB interromperà la restituzione di documenti dopo aver restituito «n» documenti, anche se ci sono più corrispondenze nel database.

`FindIterable`supporta sia gli iteratori che i modelli di cursore durante il recupero di documenti da Amazon DocumentDB. Il vantaggio dell'utilizzo `FindIterable` come iteratore è che consente il caricamento lento dei documenti e recupera i documenti solo quando richiesto dall'applicazione. Un altro vantaggio dell'utilizzo di iterator è che non sei responsabile del mantenimento della connessione al cluster e quindi non è richiesta una chiusura esplicita della connessione.

`FindIterable`forniscono anche un supporto `MongoCursor` che consente di utilizzare i pattern dei cursori quando si lavora con le query di Amazon DocumentDB. `MongoCursor`è un'implementazione specifica del driver Java di MongoDB che fornisce il controllo sulle operazioni del database e sulla gestione delle risorse. Implementa l'`AutoCloseable`interfaccia, consentendo una gestione esplicita delle risorse tramite blocchi try-with-resources, fondamentale per chiudere correttamente le connessioni al database e liberare le risorse del server. Per impostazione predefinita, il cursore scade dopo 10 minuti e DocumentDB non offre la possibilità di modificare questo comportamento di timeout. Quando lavori con dati in batch, assicurati di recuperare il batch di dati successivo prima che scada il

timeout del cursore. Una considerazione fondamentale nell'utilizzo `MongoCursor` è che richiede una chiusura esplicita per evitare perdite di risorse.

In questa sezione, vengono presentati diversi esempi per`find()`, e`Filters`. `FindIterable`

Il seguente esempio di codice mostra come utilizzare per `find()` recuperare un singolo documento utilizzando il relativo campo «restaurantID»:

```
Document filter = new Document("restaurantId", "REST-21G145");
Document result = collection.find(filter).first();
```

Sebbene l'utilizzo `Filters` consenta un migliore controllo degli errori in fase di compilazione, il driver java consente anche di specificare un `Bson` filtro direttamente nel metodo. `find()` Il codice di esempio seguente passa il `Bson` documento a`find()`:

```
result = collection.find(new Document("$and", Arrays.asList(
    new Document("rating.totalReviews", new Document("$gt", 1000)),
    new Document("priceRange", "$$"))))
```

Il codice di esempio successivo mostra diversi esempi di utilizzo della `Filters` classe con`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));
```

L'esempio seguente mostra come concatenare le operazioni di `sort()skip()`,`limit()`, e `batchSize()` su un `FindIterable` oggetto. L'ordine in cui vengono fornite queste operazioni influirà sulle prestazioni della query. Come procedura consigliata, l'ordine di queste operazioni dovrebbe essere `sort()projection()`,`skip()`, `limit()` e`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);
```

Il codice di esempio seguente mostra la creazione di un iteratore su`FindIterable`. Utilizza il `forEach` costrutto di Java per attraversare il set di risultati.

```
collection.find(Filters.eq("cuisine", "American")).forEach(doc ->
 System.out.println(doc.toJson()));
```

Nell'ultimo esempio di `find()` codice, mostra come utilizzare `cursor()` per il recupero dei documenti. Crea il cursore nel blocco try che assicura che il cursore venga chiuso quando il codice esce dal blocco try.

```
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
```

Aggiornamento dei documenti esistenti in una raccolta DocumentDB

Amazon DocumentDB offre meccanismi flessibili e potenti per modificare i documenti esistenti e inserirne di nuovi quando non esistono. Il driver Java MongoDB offre diversi metodi per gli aggiornamenti: per l'aggiornamento di singoli documenti, `updateOne()` per l'aggiornamento di più documenti e `updateMany()` `replaceOne()` per la sostituzione completa dei documenti. Oltre a questi tre metodi, `Updates` `UpdateOptions`, `UpdateResult` ci sono altri componenti fondamentali che forniscono gli elementi costitutivi per le operazioni di aggiornamento nei driver Java MongoDB.

La `Updates` classe nel driver Java MongoDB è una classe di utilità che fornisce metodi di fabbrica statici per la creazione di operatori di aggiornamento. Serve come generatore principale per la creazione di operazioni di aggiornamento in modo leggibile e sicuro per i tipi. Metodi di base come `set()``unset()`, e `inc()` consentono la modifica diretta dei documenti. La potenza di questa classe diventa evidente quando si combinano più operazioni utilizzando il `Updates.combine()` metodo che consente di eseguire più operazioni di aggiornamento in modo atomico, garantendo la coerenza dei dati.

`UpdateOptions`è una potente classe di configurazione nel driver Java di MongoDB che fornisce funzionalità di personalizzazione essenziali per le operazioni di aggiornamento dei documenti. Due aspetti importanti di questa classe sono il supporto per upsert e array filter per le operazioni di

aggiornamento. La funzionalità upsert, abilitata tramite`upsert(true)`, consente la creazione di nuovi documenti quando non viene trovato alcun documento corrispondente durante un'operazione di aggiornamento. Tramite`arrayFilters()`, l'operazione di aggiornamento può aggiornare con precisione gli elementi dell'array che soddisfano criteri specifici.

`UpdateResult`nel driver Java di MongoDB fornisce il meccanismo di feedback che descrive in dettaglio l'esito di un'operazione di aggiornamento. Questa classe racchiude tre metriche chiave: il numero di documenti che corrispondono ai criteri di aggiornamento (`matchedCount`), il numero di documenti effettivamente modificati () e le informazioni su eventuali documenti modificati (`modifiedCount`). `upsertedId` La comprensione di queste metriche è essenziale per una corretta gestione degli errori, la verifica delle operazioni di aggiornamento e il mantenimento della coerenza dei dati nelle applicazioni.

Aggiorna e sostituisci un singolo documento

In DocumentDB, l'aggiornamento di un singolo documento può essere eseguito utilizzando il metodo updateOne (). Questo metodo utilizza un parametro di filtro, in genere fornito dalla `Filters` classe, per identificare il documento da aggiornare, un parametro `Updat` e che determina i campi da aggiornare e un `UpdateOptions` parametro opzionale per impostare diverse opzioni per l'aggiornamento. L'utilizzo del `updateOne()` metodo aggiornerà solo il primo documento che corrisponde ai criteri di selezione. Il codice di esempio seguente aggiorna un singolo campo di un documento:

```
collection.updateOne(Filters.eq("restaurantId", "REST-Y2E9H5"),
    Updates.set("name", "Amazing Japanese sushi"));
```

Per aggiornare più campi in un documento, utilizzare `updateOne()` with `Update.combine()` come mostrato nell'esempio seguente. Questo esempio mostra anche come aggiungere un elemento a un array nel documento.

```
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);
```

L'esempio di codice seguente dimostra come aggiornare un documento nel database. Se il documento specificato non esiste, l'operazione lo inserirà automaticamente come nuovo documento. Questo codice mostra anche come utilizzare le metriche disponibili tramite l'`UpdateResult`oggetto.

```
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)");
}
```

L'esempio di codice seguente dimostra come sostituire completamente un documento esistente con un nuovo documento utilizzando il `replaceOne()` metodo, anziché aggiornare i singoli campi. Il `replaceOne()` metodo sovrascrive l'intero documento, preservando solo il `_id` campo dell'originale. Se più documenti soddisfano i criteri del filtro, viene sostituito solo il primo documento rilevato.

```
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());
```

Aggiornare più documenti

Esistono due modi per aggiornare più documenti in una raccolta contemporaneamente. È possibile utilizzare il `updateMany()` metodo o utilizzare il metodo `UpdateManyModel` con il `bulkWrite()`

metodo. Il `updateMany()` metodo utilizza un parametro di filtro per selezionare i documenti da aggiornare, il `Update` parametro per identificare i campi da aggiornare e un `UpdateOptions` parametro opzionale per specificare le opzioni di aggiornamento.

Il codice di esempio seguente dimostra l'utilizzo del `updateMany()` metodo:

```
Bson filter = Filters.and(
    Filters.in("features", Arrays.asList("Private Dining")),
    Filters.eq("cuisine", "Thai"));
UpdateResult result1 = collection.updateMany(filter, Updates.set("priceRange", "$$$"));
```

Il codice di esempio seguente illustra il `bulkWrite()` metodo che utilizza lo stesso aggiornamento:

```
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());
```

Rimozione di documenti da una raccolta DocumentDB

Il driver Java MongoDB consente `deleteOne()` di rimuovere un singolo documento e `deleteMany()` di rimuovere più documenti che corrispondono a criteri specifici. Proprio come l'aggiornamento, anche l'operazione di eliminazione può essere utilizzata con il `bulkWrite()` metodo. Entrambi `deleteOne()` `deleteMany()` restituiscono un `DeleteResult` oggetto che fornisce informazioni sull'esito dell'operazione, incluso il conteggio dei documenti eliminati. Di seguito è riportato un esempio di utilizzo `deleteMany()` per rimuovere più documenti:

```
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());
```

Gestione degli errori con logica di riprova

Una solida strategia di gestione degli errori per Amazon DocumentDB dovrebbe implementare la categorizzazione degli errori in ripetibili (come timeout di rete, problemi di connessione) e non ripetibili (come errori di autenticazione, query non valide). In caso di errori operativi dovuti a errori che devono essere ripetuti, dovrebbe implementare un ritardo tra ogni tentativo e il numero massimo di tentativi di ripetizione. Le operazioni CRUD dovrebbero trovarsi in un blocco try-catch contenente catches e le relative sottoclassi. `MongoException` Inoltre, dovrebbe includere il monitoraggio e la registrazione degli errori per la visibilità operativa. Di seguito è riportato un esempio di codice che mostra come implementare la gestione degli errori relativi ai nuovi tentativi:

```
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);
    }
}
```

Gestione degli indici in Amazon DocumentDB con Java

Gli indici consentono un recupero efficiente dei dati da una raccolta Amazon DocumentDB. Senza indici, DocumentDB deve scansionare tutti i documenti della raccolta per restituire risultati che soddisfino una determinata query. Questo argomento fornisce informazioni su come creare, eliminare

ed elencare gli indici utilizzando i driver Java MongoDB. Viene inoltre descritto come determinare se un determinato indice viene utilizzato nella query e come fornire suggerimenti ad Amazon DocumentDB per utilizzare un indice specifico.

Argomenti

- Creazione di indici con Java
- Eliminazione degli indici
- Determinazione della selezione dell'indice e fornitura di un suggerimento sull'indice

Amazon DocumentDB supporta molti tipi di indici. Per una panoramica completa di tutti gli indici supportati, consulta Come indicizzare su Amazon DocumentDB nel blog del database. AWS

Creazione di indici con Java

Esistono due meccanismi per creare indici in Amazon DocumentDB utilizzando i driver Java MongoDB: tramite e tramite `runCommand()` il `createIndex()` metodo per un singolo indice o il metodo per più indici. `createIndexes()` Uno dei motivi per utilizzare i `createIndexes()` metodi `createIndex()` and è che puoi migliorare la gestione degli errori rilevando errori specifici relativi alla creazione dell'indice. Un altro motivo per utilizzare questi metodi `runCommand()` è che il driver Java MongDB fornisce un ricco set di classi di supporto per la creazione e la manipolazione degli indici. Nota che queste classi di supporto possono essere utilizzate solo quando si utilizzano i `createIndex()` metodi or. `createIndexes()` Esistono tre classi di supporto:

- **`Indexes`**— Questa classe funge da classe di utilità che offre metodi di fabbrica statici per la creazione di vari tipi di indici. Semplifica il processo di creazione di definizioni di indici complesse ed è comunemente usata insieme ad altre classi relative agli indici.
- **`IndexModel`**— Questa è una classe fondamentale che racchiude sia la definizione delle chiavi dell'indice che le relative opzioni. Rappresenta una specifica completa dell'indice, che combina cosa indicizzare (le chiavi) con come indicizzare (le opzioni). Questa classe è particolarmente utile quando si creano più indici contemporaneamente, in quanto consente di definire una raccolta di specifiche di indice che possono essere passate al `createIndexes()` metodo.
- **`IndexOptions`**— Questa è una classe di configurazione completa che fornisce un ricco set di metodi per personalizzare il comportamento degli indici. Include impostazioni per indici univoci, indici sparsi, tempo di scadenza (TTL) ed espressioni di filtro parziali. Tramite il concatenamento di metodi, è possibile configurare più opzioni come la creazione di indici in background e vincoli univoci.

Crea un indice singolo

Questo esempio mostra come creare un indice singolo utilizzando il metodo`createIndex()` in background. Per informazioni sulla creazione di indici in background e in primo piano, fare riferimento a. Tipi di compilazione dell'indice Il seguente esempio di codice viene utilizzato `IndexOptions` per creare un indice univoco con il nome «unique_restaurantId_idx» in background. Questo `IndexOptions` oggetto viene quindi passato al metodo. `createIndex()`

```
collection.createIndex(
    Indexes.ascending("restaurantId"),
    new IndexOptions()
        .unique(true)
        .name("unique_restaurantId_idx")
        .background(true));
```

Crea più indici

Questo esempio crea più indici utilizzando il metodo. `createIndexes()` Per prima cosa crea l'opzione per ogni indice utilizzando l'`IndexModel`oggetto e quindi passa un elenco di `IndexModel` oggetti al metodo. `createIndexes()` L'esempio di codice seguente mostra come creare un indice composito utilizzando la classe di `Indexes` utilità. Questa classe viene utilizzata anche per specificare se si desidera creare un indice utilizzando un ordinamento crescente o decrescente. Dopo aver creato più indici, verifica la creazione dell'indice chiamando il metodo. `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()));
```

Crea indici sparsi e parziali

Questo esempio mostra la creazione di un indice sparso e parziale creando un indice `IndexModel` per ogni tipo di indice.

```
// 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))));
```

Crea un indice di testo

Questo esempio mostra come creare un indice di testo. In una raccolta è consentito un solo indice di testo, ma tale indice di testo può essere un indice composto che copre più campi. Quando si utilizzano più campi nell'indice di testo, è anche possibile assegnare pesi a ciascuno dei campi dell'indice. Gli indici di testo sui campi di matrice non sono supportati da Amazon DocumentDB e anche se è possibile utilizzare fino a 30 campi nell'indice di testo composto, è possibile assegnare un peso solo a tre campi.

```
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());
```

Crea un indice usando **`runCommand()`**

Amazon DocumentDB supporta la creazione parallela di indici per ridurre il tempo necessario per la creazione degli indici. L'indicizzazione parallela utilizza più lavoratori simultanei. I worker predefiniti utilizzati per la creazione dell'indice sono due. Per ulteriori informazioni, consulta Unlock the power of parallel indexing in Amazon DocumentDB nel Database Blog. AWS Attualmente, i driver Java MongDB non supportano la specificazione dell'opzione worker quando si utilizza `createIndex()` or `createIndexes()` e quindi l'unico modo per specificare i worker è tramite. `runCommand` Il seguente esempio di codice dimostra come utilizzare per `runCommand` creare un indice che aumenti il lavoratore a quattro:

```
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);
```

Eliminazione degli indici

Il driver Java MongoDB offre diversi metodi per eliminare gli indici, in base a diversi scenari e preferenze. Puoi eliminare gli indici per nome, per specifica della chiave o eliminare tutti gli indici contemporaneamente. I metodi `dropIndex()` `dropIndexes()` possono essere richiamati su un oggetto di raccolta per eliminare un indice. Quando si elimina un indice per nome, è necessario assicurarsi che utilizzino il nome di indice corretto, il che potrebbe non essere sempre intuitivo, specialmente per gli indici composti o generati automaticamente. Il tentativo di eliminare un indice

inesistente comporterà un. MongoCommandException L'`default _id`indice non può essere eliminato poiché garantisce l'unicità del documento all'interno della raccolta.

L'esempio di codice seguente mostra come eliminare un indice fornendo il nome del campo in cui viene creato l'indice o eliminando tutti gli indici:

```
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();
```

Quando si eliminano gli indici utilizzando più chiavi, assicurati che sia presente un indice composto contenente tutte le chiavi specificate e che l'ordine delle chiavi sia corretto. Il codice di esempio di creazione dell'indice riportato sopra mostra una chiave composta su «cucina» e caratteristiche. Se si tenta di eliminare quella chiave composta ma l'ordine non è quello utilizzato per la creazione, si verifica un MongoCommnadException errore come segue:

```
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);
}
```

Viene visualizzato l'errore seguente:

```
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
```

Determinazione della selezione dell'indice e fornitura di un suggerimento sull'indice

Utilizzare la funzionalità di spiegazione di Amazon DocumentDB è fondamentale per comprendere le prestazioni delle query e l'utilizzo degli indici. Quando si esegue una query, è possibile aggiungere il `explain()` metodo per ottenere informazioni dettagliate sul piano di query, inclusi gli eventuali indici utilizzati. L'`explain()`output fornisce informazioni dettagliate sulle fasi di esecuzione della query, sul numero di documenti esaminati e sul tempo impiegato per ciascuna fase. Queste informazioni sono preziose per identificare se un determinato indice viene utilizzato in modo efficace o se la query potrebbe trarre vantaggio da una diversa struttura dell'indice.

Il `explain()` metodo può essere concatenato al `find()` metodo. Il `explain()` metodo può accettare un'`ExplainVerbosity`enumerazione opzionale che determina il livello di verbosità restituito da. `explain()` Al momento, solo gli `QUERY_PLANNER` enumeratori `EXECUTION_STATS` e sono supportati da DocumentDB. Il seguente esempio di codice mostra come ottenere Query Planner per una query specifica:

```
// 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());
```

Il seguente documento JSON viene restituito per il livello di verbosità del Query Planner:

```
{
  "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
    }
  }
}
```

Sono disponibili diverse opzioni per influenzare o forzare Amazon DocumentDB a utilizzare un indice specifico. `hintString()`I metodi `hint()` and consentono di sovrascrivere il comportamento di selezione dell'indice predefinito di Query Optimizer specificando esplicitamente quale indice deve essere utilizzato per una query. Sebbene l'ottimizzatore di query di DocumentDB sia generalmente una buona scelta per la selezione dell'indice, ci sono scenari in cui forzare `hint()` o `hintString()` può essere utile forzare un indice specifico, ad esempio quando si hanno a che fare con dati distorti o si testano le prestazioni dell'indice.

Il seguente esempio di codice impone l'uso dell'indice composto «cuisine_features_idx» per la stessa query eseguita nel codice precedente:

```
// 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 programmazione con Amazon DocumentDB e Java

Event-driven la programmazione nel contesto di Amazon DocumentDB rappresenta un potente modello architetturale in cui le modifiche al database fungono da generatori di eventi principali che attivano la logica e i processi aziendali successivi. Quando i record vengono inseriti, aggiornati o eliminati in una raccolta DocumentDB, queste modifiche agiscono come eventi che avviano automaticamente vari processi, notifiche o attività di sincronizzazione dei dati a valle. Questo modello

è particolarmente utile nei moderni sistemi distribuiti in cui più applicazioni o servizi devono reagire alle modifiche dei dati in tempo reale. Il meccanismo principale per implementare la programmazione basata sugli eventi in DocumentDB è costituito dai flussi di modifiche.

> **Note**
>
> Questa guida presuppone che tu abbia abilitato i flussi di modifica su una raccolta con cui stai lavorando. Consulta Utilizzo dei flussi di modifiche con Amazon DocumentDB per scoprire come abilitare i flussi di modifica sulla raccolta.

Utilizzo dei flussi di modifica dall'applicazione Java

Il `watch()` metodo del driver Java di MongoDB è il meccanismo principale per monitorare le modifiche ai dati in tempo reale in Amazon DocumentDB. Il `watch()` metodo può essere richiamato da `MongoClient` `MongoDatabase`, e oggetti. `MongoCollection`

Il `watch()` metodo restituisce un'istanza `ChangeStreamIterable` che supporta varie opzioni di configurazione, tra cui la ricerca completa del documento per gli aggiornamenti, la fornitura di token di curriculum e timestamp per l'affidabilità e le fasi di aggregazione della pipeline per filtrare le modifiche.

`ChangeStreamIterable`implementa l'interfaccia Java principale e può essere utilizzato con. `Iterable` `forEach()` Per catturare gli eventi utilizzando`forEach()`, passate una funzione di callback `forEach()` che elabori l'evento modificato. Il seguente frammento di codice mostra come aprire un flusso di modifica su una raccolta per avviare il monitoraggio degli eventi di modifica:

```
ChangeStreamIterable < Document > iterator = collection.watch();
iterator.forEach(event - > {
    System.out.println("Received a change: " + event);
});
```

Un altro modo per visualizzare tutti gli eventi di modifica consiste nell'aprire un cursore che mantiene una connessione al cluster e riceve continuamente nuovi eventi di modifica non appena si verificano. Per ottenere un cursore dei flussi di modifica, utilizzate il metodo dell'oggetto. `cursor()` `ChangeStreamIterable` Il seguente esempio di codice mostra come monitorare gli eventi di modifica utilizzando il cursore:

```
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().cursor()) {
```

```
    System.out.println(cursor.tryNext());
}
```

Come procedura consigliata, è possibile creare l'istruzione `MongoChangeStreamCursor` in a try-with-resource o chiudere manualmente il cursore. La chiamata al `cursor()` metodo on `ChangeStreamIterable` restituisce un elemento creato su un `MongoChangeStreamCursor` oggetto. `ChangeStreamDocument`

La `ChangeStreamDocument` classe è un componente cruciale che rappresenta i singoli eventi di modifica nello stream. Contiene informazioni dettagliate su ciascuna modifica, tra cui il tipo di operazione (inserimento, aggiornamento, eliminazione, sostituzione), la chiave del documento, le informazioni sullo spazio dei nomi e il contenuto completo del documento, se disponibile. La classe fornisce metodi per accedere a vari aspetti dell'evento di modifica, `getOperationType()` ad esempio per determinare il tipo di modifica, accedere `getFullDocument()` allo stato completo del documento e `getDocumentKey()` identificare il documento modificato.

L'`ChangeStreamDocument`oggetto fornisce due informazioni importanti, un token di curriculum e l'ora dell'evento di modifica.

I token di curriculum e le operazioni basate sul tempo nei flussi di modifica di DocumentDB forniscono meccanismi cruciali per mantenere la continuità e gestire la cronologia degli accessi ai dati. Un token di curriculum è un identificatore univoco generato per ogni evento di modifica, che funge da segnalibro che consente alle applicazioni di riavviare l'elaborazione del flusso di modifiche da un punto specifico dopo disconnessioni o errori. Quando viene creato un cursore del flusso di modifiche, può utilizzare un token di curriculum precedentemente memorizzato tramite l'`resumeAfter()`opzione, consentendo allo stream di continuare dal punto in cui era stato interrotto anziché ricominciare dall'inizio o perdere eventi.

Time-based le operazioni nei flussi di modifica offrono approcci diversi per gestire il punto di partenza del monitoraggio degli eventi di modifica. L'`startAtOperationTime()`opzione consente di iniziare a guardare le modifiche avvenute in corrispondenza o dopo un determinato timestamp. Queste funzionalità basate sul tempo sono particolarmente utili in scenari che richiedono l'elaborazione di dati storici, il ripristino point-in-time o la sincronizzazione tra sistemi.

L'esempio di codice seguente recupera l'evento associato al documento di inserimento, ne acquisisce il token di curriculum e quindi fornisce quel token per avviare il monitoraggio degli eventi successivi all'evento di inserimento. L'evento è associato all'evento di aggiornamento, quindi ottiene l'ora del cluster in cui è avvenuto l'aggiornamento e utilizza tale timestamp come punto di partenza per ulteriori elaborazioni.

```
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());
  }
```

Per impostazione predefinita, l'evento di modifica dell'aggiornamento non include il documento completo e include solo le modifiche apportate. Se è necessario accedere al documento completo che è stato aggiornato, è possibile chiamare il `fullDocument()` metodo sull'`ChangeStreamIterable`oggetto. Tieni presente che quando richiedi la restituzione di un documento completo per un evento di aggiornamento, viene restituito il documento esistente nel momento in cui viene effettuata la chiamata per modificare i flussi di modifica.

Questo metodo accetta un `FullDocument` enum come parametro. Attualmente, Amazon DocumentDB supporta solo DEFAULT e `UPDATE_LOOKUP` valori. Il seguente frammento di codice mostra come richiedere il documento completo per gli eventi di aggiornamento quando si inizia a controllare le modifiche:

```
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().fullDocument(FullDocument.UPDATE_LOOKUP).cursor())
```

Utilizzo della convalida dello schema JSON

Utilizzando l'operatore di interrogazione di `$jsonSchema` valutazione, puoi convalidare i documenti inseriti nelle tue raccolte.

Argomenti

- Creazione e utilizzo della convalida dello schema JSON
- Parole chiave supportate
- bypassare DocumentValidation
- Limitazioni

Creazione e utilizzo della convalida dello schema JSON

Creazione di una raccolta con convalida dello schema

È possibile creare una raccolta con regole `createCollection` operative e di convalida. Queste regole di convalida vengono applicate durante gli inserimenti o gli aggiornamenti dei documenti Amazon DocumentDB. Il seguente esempio di codice mostra le regole di convalida per un insieme di dipendenti:

```
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"
} )
```

Inserimento di un documento valido

L'esempio seguente inserisce documenti conformi alle regole di convalida dello schema di cui sopra:

```
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})
```

Inserimento di un documento non valido

L'esempio seguente inserisce documenti che non sono conformi alle regole di convalida dello schema di cui sopra. In questo esempio, il valore EmployeeID non è una stringa:

```
db.employees.insert({
    "name" : { "firstName" : "Carol" , "lastName" : "Smith"},
    "employeeId": 720 ,
    "salary": 1000.0
})
```

Questo esempio mostra una sintassi errata all'interno del documento.

Modifica di una raccolta

Il `collMod` comando viene utilizzato per aggiungere o modificare le regole di convalida della raccolta esistente. L'esempio seguente aggiunge un campo stipendio all'elenco dei campi obbligatori:

```
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
      }
   }
} )
```

Indirizzare i documenti aggiunti prima della modifica delle regole di convalida

Per indirizzare i documenti che sono stati aggiunti alla tua raccolta prima della modifica delle regole di convalida, utilizza i seguenti `validationLevel` modificatori:

- strict: applica le regole di convalida a tutti gli inserti e gli aggiornamenti.

- moderate: applica le regole di convalida ai documenti validi esistenti. Durante gli aggiornamenti, i documenti non validi esistenti non vengono controllati.

Nell'esempio seguente, dopo aver aggiornato le regole di convalida sulla raccolta denominata «dipendenti», il campo stipendio è obbligatorio. L'aggiornamento del seguente documento avrà esito negativo:

```
db.runCommand({
    update: "employees",
    updates: [{
        q: { "employeeId": "c721a" },
        u: { age: 25 , salary : 1000},
        upsert: true }]
})
```

Amazon DocumentDB restituisce il seguente output:

```
{
"n" : 0,
    "nModified" : 0,
    "writeErrors" : [
        {
"index" : 0,
            "code" : 121,
            "errmsg" : "Document failed validation"
        }
    ],
    "ok" : 1,
    "operationTime" : Timestamp(1234567890, 1)
}
```

L'aggiornamento del livello di convalida per `moderate` consentire l'aggiornamento corretto del documento precedente:

```
db.runCommand({
    "collMod" : "employees",
    validationLevel : "moderate"
})

db.runCommand({
    update: "employees",
```

```
    updates: [{
        q: { "employeeId": "c721a" },
        u: { age: 25 , salary : 1000},
        upsert: true }]
})
```

Amazon DocumentDB restituisce il seguente output:

```
{
"n" : 1,
    "nModified" : 1,
    "ok" : 1,
    "operationTime" : Timestamp(1234567890, 1)
}
```

Recupero di documenti con $jsonSchema

L'`$jsonSchema`operatore può essere utilizzato come filtro per interrogare i documenti che corrispondono allo schema JSON. Si tratta di un operatore di primo livello che può essere presente nei documenti di filtro come campo di primo livello o utilizzato con operatori di query come `$and` `$or`, e. `$nor` Gli esempi seguenti mostrano l'uso di $JsonSchema come filtro singolo e con altri operatori di filtro:

Documento inserito in una raccolta di «dipendenti»:

```
{ "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 }
```

Raccolta filtrata solo con l'`$jsonSchema`operatore:

```
db.employees.find({
       $jsonSchema: { required: ["age"] } })
```

Amazon DocumentDB restituisce il seguente output:

```
{ "_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 }
```

Raccolta filtrata con l'`$jsonSchema`operatore e un altro operatore:

```
db.employees.find({
       $or: [{ $jsonSchema: { required: ["age", "name"]}},
            { salary: { $lte:1000}}]});
```

Amazon DocumentDB restituisce il seguente output:

```
{ "_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 }
```

Raccolta filtrata con l'`$jsonSchema`operatore e all'`$match`interno del filtro aggregato:

```
db.employees.aggregate(
    [{ $match: {
        $jsonSchema: {
            required: ["name", "employeeId"],
            properties: {"salary" :{"bsonType": "double"}}
        }
       }
    }]
)
```

Amazon DocumentDB restituisce il seguente output:

```
{
"_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
}
```

Visualizzazione delle regole di convalida esistenti

Per visualizzare le regole di convalida esistenti su una raccolta, usa:

```
db.runCommand({
    listCollections: 1,
    filter: { name: 'employees' }
})
```

Amazon DocumentDB restituisce il seguente output:

```
{
    "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 mantiene anche le regole di convalida nella fase di aggregazione. `$out`

Parole chiave supportate

I seguenti campi sono supportati nei `collMod` comandi `create` and:

- **`Validator`**— Supporta l'operatore `$jsonSchem` a.
- **`ValidationLevel`**— Supporti `off` `strict` e `moderate` valori.
- **`ValidationAction`**— Supporta il `error` valore.

L'operatore $JsonSchema supporta le seguenti parole chiave:

- `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`

bypassare DocumentValidation

Amazon DocumentDB supporta `bypassDocumentValidation` i seguenti comandi e metodi:

- `insert`
- `update`
- `findAndModify`
- `$outstage` nel `aggregate` comando e nel metodo `db.collection.aggregate()`

Amazon DocumentDB non supporta i seguenti comandi per: `bypassDocumentValidation`

- `$merge`nel `aggregate` comando e nel metodo `db.collection.aggregate()`
- `mapReduce`comando e `db.collection.mapReduce()` metodo
- Comando `applyOps`

Limitazioni

Le seguenti limitazioni si applicano alla `$jsonSchema` convalida:

- Amazon DocumentDB restituisce l'errore «Document failed validation» quando un'operazione non soddisfa la regola di convalida.
- I cluster elastici di Amazon DocumentDB non sono supportati. `$jsonSchema`

Interrogazione in Amazon DocumentDB

Questa sezione spiega tutti gli aspetti delle interrogazioni con Amazon DocumentDB.

Argomenti

- Interrogazione di documenti
- Piano di query
- Spiegare i risultati
- Pianificatore di query v2
- Pianificatore di query v3
- Interrogazione di dati geospaziali con Amazon DocumentDB
- Indice parziale
- Esecuzione di ricerche di testo con Amazon DocumentDB

Interrogazione di documenti

A volte, potrebbe essere necessario cercare nell'inventario dello store online, in modo che i clienti possano vedere e acquistare ciò che vendi. Eseguire query su una raccolta è relativamente semplice, sia per le ricerche su tutti i documenti della raccolta che per quelle solo sui documenti che soddisfano un determinato criterio.

Per eseguire una query per i documenti, utilizza l'operazione `find()`. Il comando `find()` ha un parametro per documenti singoli che definisce i criteri da utilizzare nella scelta dei documenti da

restituire. L'output di `find()` è un documento formattato su una sola riga di testo senza interruzioni di riga. Per formattare il documento di output per facilitare la lettura, utilizza `find().pretty()`. Tutti gli esempi di questo argomento utilizzano `.pretty()` per formattare l'output.

I seguenti esempi di codice utilizzano i quattro documenti che avete inserito nella `example` raccolta nei due esercizi precedenti `insertOne()` e `insertMany()` che si trovano in. Aggiungere documenti

Argomenti

- Recupero di tutti i documenti di una raccolta
- Recupero di documenti che corrispondono a un campo Valore
- Recupero di documenti che corrispondono a un documento incorporato
- Recupero di documenti che corrispondono al valore di un campo in un documento incorporato
- Recupero di documenti che corrispondono a un array
- Recupero di documenti che corrispondono a un valore in un array
- Recupero di documenti mediante operatori

Recupero di tutti i documenti di una raccolta

Per recuperare tutti i documenti nella raccolta, usa l'operazione `find()` con un documento di query vuoto.

La query seguente restituisce tutti i documenti della raccolta `example`.

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

Recupero di documenti che corrispondono a un campo Valore

Per recuperare tutti i documenti che corrispondono a un campo e a un valore, usa l'operazione `find()` con un documento di query che identifica i campi e i valori per la corrispondenza.

Usando i documenti precedenti, questa query restituisce tutti i documenti in cui il campo "Item" è uguale a "Pen".

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

Recupero di documenti che corrispondono a un documento incorporato

Per trovare tutti i documenti che corrispondono a un documento incorporato, utilizza l'operazione `find()` con un documento di query che specifica il nome del documento incorporato e tutti i campi e i valori per quel documento incorporato.

Quando si esegue il confronto di un documento incorporato, il documento incorporato del documento deve avere lo stesso nome che ha nella query. Inoltre, i campi e i valori nel documento incorporato devono corrispondere alla query.

La seguente query restituisce solo il documento "Poster Paint". Questo perché "Pen" ha valori diversi per "`OnHand`" e "`MinOnHand`" e "Spray Paint" ha un ulteriore campo (`OrderQnty`) rispetto al documento di query.

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

Recupero di documenti che corrispondono al valore di un campo in un documento incorporato

Per trovare tutti i documenti che corrispondono a un documento incorporato, utilizza l'operazione `find()` con un documento di query che specifica il nome del documento incorporato e tutti i campi e i valori per quel documento incorporato.

Considerati i documenti precedenti, la seguente query utilizza la "dot notation" (notazione col punto) per specificare il documento incorporato e i campi di interesse. Vengono restituiti tutti i documenti che corrispondono a questi campi, indipendentemente da quali altri campi possono essere presenti nel documento incorporato. La query restituisce "Poster Paint" e "Spray Paint" perché entrambi corrispondono ai campi e ai valori specificati.

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

Recupero di documenti che corrispondono a un array

Per trovare tutti i documenti che corrispondono a una matrice, utilizzare l'operazione `find()` con il nome della matrice richiesta e tutti i valori in quella matrice. La query restituisce tutti i documenti che hanno una matrice con quel nome in cui i valori della matrice sono identici e nello stesso ordine rispetto alla query.

La seguente query restituisce solo "Pen" perché" Poster Paint" ha un ulteriore colore (White) mentre "Spray Paint" ha i colori in un ordine diverso.

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

Recupero di documenti che corrispondono a un valore in un array

Per trovare tutti i documenti che hanno un valore specifico di matrice, utilizza l'operazione `find()` con il nome della matrice e il valore richiesto.

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

L'operazione precedente restituisce tutti e tre i documenti, in quanto ciascuno di essi dispone di una matrice denominata `Colors` e del valore "`Red`" all'interno della matrice. Se si specifica il valore "`White`", la query restituisce solo "Poster Paint".

Recupero di documenti mediante operatori

La seguente query restituisce tutti i documenti in cui il valore "`Inventory.OnHand`" è inferiore a 50.

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

Per un elenco degli operatori di query supportati, consulta Operatori di interrogazione e proiezione.

Piano di query

Come posso vedere gli **`ExecutionStats`** per un piano di interrogazione?

Quando si determina il motivo per cui una query viene eseguita più lentamente del previsto, può essere utile capire quali sono i valori `executionStats` per il piano di query. `executionStats` Fornisce il numero di documenti restituiti da una particolare fase (`nReturned`), la quantità di tempo di esecuzione trascorso in ogni fase (`executionTimeMillisEstimate`) e la quantità di tempo necessaria per generare un piano di query (`planningTimeMillis`). È possibile determinare le fasi della query che richiedono più tempo per concentrare gli sforzi di ottimizzazione a partire dall'output di`executionStats`, come mostrato nei seguenti esempi di query. Il parametro `executionStats` attualmente non supporta i comandi `update` e `delete`.

> **Note**
>
> Amazon DocumentDB emula l'API MongoDB 3.6 su un motore di database appositamente progettato che utilizza un sistema di storage distribuito, tollerante ai guasti e con risoluzione automatica. Di conseguenza, i piani di interrogazione e l'output di possono differire tra Amazon DocumentDB e MongoDB. `explain()` I clienti che desiderano il controllo sul piano di query possono utilizzare l'operatore `$hint` per applicare la selezione di un indice preferito.

Esegui la query che vuoi ottimizzare con il comando `explain()` come segue.

```
db.runCommand({explain: {query document}}).
explain("executionStats").executionStats;
```

Di seguito è riportato un esempio di operazione.

```
db.fish.find({}).limit(2).explain("executionStats");
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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
}
```

Se si desidera vedere solo il valore `executionStats` dalla query di cui sopra, è possibile utilizzare il seguente comando. Per piccole raccolte, il processore di query Amazon DocumentDB può scegliere di non utilizzare un indice se i miglioramenti delle prestazioni sono trascurabili.

```
db.fish.find({}).limit(2).explain("executionStats").executionStats;
```

Cache del piano di query

Per ottimizzare le prestazioni e ridurre la durata della pianificazione, Amazon DocumentDB memorizza internamente i piani di query nella cache. Ciò consente di eseguire query con la stessa forma direttamente utilizzando un piano memorizzato nella cache.

Tuttavia, questa memorizzazione nella cache può talvolta causare un ritardo casuale per la stessa query; ad esempio, una query che in genere richiede un secondo per essere eseguita può occasionalmente richiedere dieci secondi. Questo perché nel tempo, l'istanza del lettore ha memorizzato nella cache varie forme della query, consumando così memoria. Se si verifica questa lentezza casuale, non è necessario eseguire alcuna azione per liberare la memoria: il sistema gestirà l'utilizzo della memoria per conto dell'utente e, una volta raggiunta una certa soglia, verrà rilasciata automaticamente.

Spiegare i risultati

Se desideri restituire informazioni sui piani di interrogazione, Amazon DocumentDB supporta la modalità di dettaglio. `queryPlanner` I `explain` risultati restituiscono il piano di query selezionato scelto dall'ottimizzatore in un formato simile al seguente:

```
{
   "queryPlanner" : {
      "plannerVersion" : <int>,
      "namespace" : <string>,
      "winningPlan" : {
         "stage" : <STAGE1>,
         ...
         "inputStage" : {
            "stage" : <STAGE2>,
            ...
            "inputStage" : {
               ...
            }
         }
      }
   }
}
```

Le sezioni seguenti definiranno `explain` risultati comuni.

Argomenti

- Fase di scansione e filtro
- Intersezione dell'indice
- Unione degli indici
- Indice multiplo intersection/union
- Indice composto
- Ordina fase
- Fase a gironi

Fase di scansione e filtro

L'ottimizzatore può scegliere una delle seguenti scansioni:

COLLSCAN

Questa fase è una scansione di raccolta sequenziale.

```
{
    "stage" : "COLLSCAN"
}
```

ISCAN

Questa fase esegue la scansione dei tasti dell'indice. L'ottimizzatore può recuperare il documento in questa fase e ciò può comportare una fase FETCH aggiunta successivamente.

```
db.foo.find({"a": 1})
{
    "stage" : "IXSCAN",
    "direction" : "forward",
    "indexName" : <idx_name>
}
```

FETCH

Se l'ottimizzatore ha recuperato i documenti in una fase diversa da IXSCAN, il risultato includerà una fase FETCH. Ad esempio, la query IXSCAN precedente può comportare una combinazione di fasi FETCH e IXSCAN:

```
db.foo.find({"a": 1})
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXSCAN",
        "indexName" : <idx_name>
    }
}
```

IXONLYSCAN analizza solo la chiave dell'indice. La creazione di indici composti non eviterà FETCH.

Intersezione dell'indice

IXAND

Amazon DocumentDB può includere uno stage IXAND con un array InputStages di IXSCAN se può utilizzare l'intersezione degli indici. Ad esempio, potremmo vedere risultati come:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXAND",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "indexName" : "a_1"
            },
            {
                "stage" : "IXSCAN",
                "indexName" : "b_1"
            }
        ]
    }
}
```

Unione degli indici

IXOR

Analogamente all'intersezione degli indici, Amazon DocumentDB può includere uno `IXOR` stage con un `inputStages` array per l'operatore. `$or`

```
db.foo.find({"$or": [{"a": {"$gt": 2}}, {"b": {"$lt": 2}}]})
```

Per la query precedente, l'output di spiegazione potrebbe essere simile al seguente:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
```

```
            {
                "stage" : "IXSCAN",
                "indexName" : "a_1"
            },
            {
                "stage" : "IXSCAN",
                "indexName" : "b_1"
            }
        ]
    }
}
```

Indice multiplo intersection/union

Amazon DocumentDB può combinare più fasi di intersezione o unione di indici e recuperare il risultato. Ad esempio:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                ...
            },
            {
                "stage" : "IXAND",
                "inputStages" : [
                    {
                        "stage" : "IXSCAN",
                        ...
                    },
                    {
                        "stage" : "IXSCAN",
                        ...
                    }
                ]
            }
        ]
    }
}
```

L'utilizzo delle fasi di intersezione o unione dell'indice non è influenzato dal tipo di indice (sparso, composto, ecc.).

Indice composto

L'utilizzo dell'indice composto di Amazon DocumentDB non è limitato nei sottoinsiemi iniziali dei campi indicizzati; può utilizzare l'indice con la parte relativa al suffisso ma potrebbe non essere molto efficiente.

Ad esempio, l'indice composto di `{ a: 1, b: -1 }` può supportare tutte e tre le seguenti query:

```
db.orders.find( { a: 1 } )
```

```
db.orders.find( { b: 1 } )
```

```
db.orders.find( { a: 1, b: 1 } )
```

Ordina fase

Se è presente un indice sulle chiavi di ordinamento richieste, Amazon DocumentDB può utilizzare l'indice per ottenere l'ordine. In tal caso, il risultato non includerà una `SORT` fase, ma piuttosto una `IXSCAN` fase. Se l'ottimizzatore preferisce un ordinamento semplice, includerà una fase come questa:

```
{
    "stage" : "SORT",
    "sortPattern" : {
        "a" : 1,
        "b" : -1
    }
}
```

Fase a gironi

Amazon DocumentDB supporta due diverse strategie di gruppo:

- `SORT_AGGREGATE`: Ordinamento aggregato su disco.

- `HASH_AGGREGATE`: Hash aggregato in memoria.

Pianificatore di query v2

Il nuovo pianificatore di query per Amazon DocumentDB (planner versione 2.0) offre funzionalità avanzate di ottimizzazione delle query e prestazioni migliorate. La versione 2.0 di Planner per Amazon DocumentDB 5.0 offre un miglioramento delle prestazioni fino a 10 volte rispetto alla versione precedente durante l'utilizzo e gli operatori con indici. `find` `update` I miglioramenti delle prestazioni derivano principalmente dall'utilizzo di piani di indicizzazione più ottimali e dall'abilitazione del supporto per la scansione degli indici per operatori come gli operatori di negazione (,) e nested. `$neq` `$nin` `$elementMatch` Le query della versione 2.0 di Planner vengono eseguite più velocemente grazie a migliori tecniche di stima dei costi, algoritmi ottimizzati e maggiore stabilità. La versione 2.0 di Planner supporta anche le API di filtro della cache del piano, il che migliora la stabilità del pianificatore. Con questa funzionalità, Amazon DocumentDB 5.0 offre ora la possibilità di scegliere tra diverse versioni di Query Planner.

Argomenti

- Prerequisiti
- Selezione della versione 2.0 del planner come pianificatore di query predefinito
- Best practice
- Limitazioni
- Miglioramenti agli operatori di ricerca e aggiornamento
- Pianifica i filtri della cache
- Potenziali differenze di comportamento tra le versioni 1.0, 2.0 e MongoDB di Planner
- La versione 2.0 di Planner colma il divario comportamentale con MongoDB

Prerequisiti

I seguenti prerequisiti si applicano alla versione 2.0 di Planner:

- La versione 2.0 di Planner è disponibile Regioni AWS ovunque sia disponibile la versione 5.0 del motore.
- Per scegliere di utilizzare la versione 2.0 come pianificatore di query predefinito, il cluster deve utilizzare la patch del motore versione 3.0.15902 o successiva di Amazon DocumentDB versione

5.0. Per i passaggi per l'aggiornamento alla patch della versione più recente del motore, consulta. Esecuzione di un aggiornamento della patch alla versione del motore di un cluster

- Per impostare la versione 2.0 di Planner come Query Planner predefinito, sono necessarie le autorizzazioni IAM per aggiornare i gruppi di parametri del cluster.
- Per Amazon DocumentDB 8.0, la versione 3.0 di planner è il pianificatore di query predefinito. Per informazioni dettagliate, vedi Pianificatore di query v3.

Selezione della versione 2.0 del planner come pianificatore di query predefinito

Utilizza i passaggi seguenti per selezionare 2.0 come pianificatore di query predefinito dalla console o dalla CLI:

- Segui i passaggi indicati Modifica dei parametri del cluster Amazon DocumentDB per modificare il gruppo di parametri del cluster.
- Per il parametro denominato 'PlannerVersion', modifica il valore in 2.0, indicando la versione 2.0 di Planner.
- Seleziona Applica immediatamente (selezionando Applica al riavvio la selezione sarà inefficace fino al successivo riavvio del cluster).

Best practice

Per ottenere i risultati previsti, utilizza le seguenti best practice quando applichi la versione 2.0 di Planner:

- In un cluster globale, seleziona lo stesso `plannerVersion` valore (1.0 o 2.0) nei gruppi di parametri del cluster per entrambe le regioni. Tieni presente che la selezione di versioni diverse del planner nelle regioni primarie e secondarie può causare comportamenti e prestazioni delle query non coerenti.
- L'aggiornamento alla versione 2.0 di Planner durante una finestra di manutenzione programmata o durante i periodi di traffico ridotto sarà il meno problematico, in quanto potrebbe verificarsi un aumento dei tassi di errore se la versione del pianificatore viene modificata quando i carichi di lavoro sono in esecuzione attiva.
- La versione 2.0 di Planner funziona in modo ottimale con la versione 5.0 della shell MongoDB.

Limitazioni

Le seguenti limitazioni si applicano alla versione 2.0 di Planner:

- La versione 2.0 di Planner non è supportata nei cluster elastici, che torneranno alla versione 1.0 di Planner.
- La versione 2.0 di Planner non è supportata per l'aggregazione e i comandi distinti, che ricadranno nella versione 1.0 di Planner.
- Su alcune forme di interrogazione non è possibile impostare un filtro della cache del piano. Per l'elenco corrente, vedereForme di interrogazione supportate.

Miglioramenti agli **operatori di ricerca e** aggiornamento

La versione 2.0 di Planner ottimizza le operazioni fondamentali tra cui`find`, `update``delete`, e `find-and-modify` comandi. Le seguenti sezioni a schede mostrano funzionalità avanzate per gli indici, nonché i miglioramenti delle prestazioni delle query con la versione 2.0 di Planner:

Enhanced index support

- La versione 2.0 di Planner aggiunge il supporto agli indici per gli operatori di negazione`$nin`, tra cui,, e `$ne``$not {eq}`, oltre a e. `$not {in}` `$type` `$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 }}}}})
```

- La versione 2.0 di Planner utilizza indici sparsi o parziali anche quando `$exists` non sono presenti nell'espressione della query.

```
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" })
```

- La versione 2.0 di Planner utilizzerà indici parziali anche quando la condizione della query non corrisponde esattamente all'espressione del filtro dell'indice parziale.

```
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}})
```

- La versione 2.0 di Planner utilizza la scansione parziale dell'indice con le query $elemMatch.

```
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}}})
```

- La versione 2.0 di Planner include il supporto per la scansione dell'indice`$regex`, senza la necessità di fornire il codice dell'applicazione. `$hint` `$regex`supporta l'indice solo nella ricerca di prefissi.

```
Sample Document: { "x": [1, 2, 3], "y": "apple" }
db.foo.createIndex({ "x": 1, "y": 1 })
db.foo.find({"y":{ $regex: "^a" }})
```

- La versione 2.0 di Planner migliora le prestazioni delle query che coinvolgono indici a più chiavi, con condizioni di uguaglianza sui campi a più chiavi.

```
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)
```

- La versione 2.0 di Planner migliora le prestazioni delle query che coinvolgono più filtri, in particolare sulle raccolte con documenti di dimensioni superiori a 8 KB.

```
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}}]})
```

- La versione 2.0 di Planner migliora le prestazioni delle query quando si utilizza l'`$in`operatore con un indice composto eliminando la fase di ordinamento.

```
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})
```

  Migliora inoltre le prestazioni delle query che utilizzano indici a più chiavi con elementi. `$in`

```
Sample Document: {"x": [1, 2, 3]}
db.foo.createIndex({"x": 1})
db.foo.find("x":{$in:[>100 elements]})
```

Query performance improvements

- La versione 2.0 di Planner migliora le prestazioni delle query che coinvolgono più filtri, in particolare sulle raccolte con documenti di dimensioni superiori a 8 KB.

```
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}}]})
```

- La versione 2.0 di Planner migliora le prestazioni delle query quando si utilizza l'`$in`operatore con un indice composto eliminando la fase di ordinamento.

```
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})
```

  Migliora anche le prestazioni delle query che utilizzano indici a più chiavi con elementi. `$in`

```
Sample Document: {"x": [1, 2, 3]}
db.foo.createIndex({"x": 1})
db.foo.find("x":{$in:[>100 elements]})
```

Pianifica i filtri della cache

La versione 2.0 di Planner aggiunge il supporto per i filtri della cache del piano, chiamati anche filtri di indice, che consentono di limitare il set di indici che il planner considera per una specifica forma di query. I filtri vengono impostati con un comando del database e applicati sul server, quindi se si verifica una regressione delle query è possibile attenuarla senza modificare il codice dell'applicazione.

Nella versione 2.0 di planner, i filtri della cache del piano si applicano ai `count` comandi `find` e e, a partire da Amazon DocumentDB 5.0.2 e 8.0.2, anche ai comandi,, e. `update` `delete` `findAndModify` I `aggregate` comandi `distinct` and richiedono la versione 3.0 di Planner.

Per ulteriori informazioni sul riferimento ai comandi, sulle forme di query supportate, sugli esempi funzionanti e su come i filtri interagiscono con i suggerimenti, consulta. Pianifica i filtri della cache

Potenziali differenze di comportamento tra le versioni 1.0, 2.0 e MongoDB di Planner

In alcuni casi limite, è possibile che la versione 2.0 di Planner produca risultati leggermente diversi dai risultati in MongoDB. Questa sezione illustra alcuni esempi di queste possibilità.

$(update) and $(projection)

- In alcuni casi, `$(update)` gli `$(projection)` operatori in MongoDB possono comportarsi diversamente dalla versione 1.0 del planner di Amazon DocumentDB. Di seguito sono riportati alcuni esempi:

```
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 versione 1.0 — Campo degli aggiornamenti 2022
  - MongoDB — Campo degli aggiornamenti 2022
  - Planner versione 2.0 — Campo degli aggiornamenti 2021

-
```
db.col.insert({x:[1,2,3]})
db.col.update({$and:[{x:1},{x:3}]},{$set:{"x.$":500}})
```

  - Planner versione 1.0 — Aggiorna in modo casuale il primo elemento corrispondente
  - MongoDB: aggiorna casualmente il primo elemento corrispondente
  - Planner versione 2.0 — Non effettua aggiornamenti

-
```
db.col.insert({x:[1,2,3]})
db.col.find()
```

  - Planner versione 1.0 — Seleziona casualmente l'elemento corrispondente
  - MongoDB — Seleziona casualmente l'elemento corrispondente
  - Planner versione 2.0 — Non effettua una selezione

-
```
db.col.insert({x:100})
db.col.update({x:100},{x:100})
```

  - Planner versione 1.0 — Modifiche al conteggio non modificate
  - MongoDB — Modifiche al conteggio non modificate
  - Planner versione 2.0 — Il conteggio nModified non cambia se aggiornato con lo stesso valore.

- Quando `$(update)` l'operatore viene utilizzato con`$setOnInsert`, la versione 1.0 di planner e MongoDB generano un errore, ma la versione 2.0 di planner no.
- La ridenominazione di un campo inesistente `$field` genera un errore nella versione 2.0 di Planner, mentre non produce aggiornamenti nella versione 1.0 di Planner e MongoDB.

Index behavior

- La versione 2.0 di Planner genera un errore quando `$hint` viene applicata con un indice non adatto, mentre la versione 1.0 di Planner e MongoDB no.

```
// 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"
}
```

- $nearnon può essere utilizzato nella versione 2.0 di Planner. $hint({“$natural”:1})

```
// 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" }
```

- Mentre MongoDB supporta la scansione completa dell'indice regex, la versione 2.0 di planner supporta la scansione dell'indice regex solo sui campi di prefisso.

```
// 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)
```

- Ci sono alcune differenze intrinseche nell'uso dei filtri della cache dei piani con la versione 2.0 di Planner rispetto a MongoDB. Sebbene la versione 2.0 di planner non supporti la specifica di «proiezione» e «confronto» con i filtri della cache del piano, MongoDB sì. Tuttavia, il filtro dell'indice MongoDB è solo in memoria e viene perso dopo il riavvio. La versione 2.0 di Planner mantiene i filtri degli indici tramite riavvii e patch.

Others

- Il formato dei log di controllo DML quando si utilizza la versione 2.0 di Planner varia leggermente da quello della versione 1.0 di Planner.

```
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}}
```

- La condizione dell'indice come parte del piano di spiegazione:

```
rs0:PRIMARY> db.col.createIndex({index1:1})
{
 "createdCollectionAutomatically" : false,
 "numIndexesBefore" : 1,
 "numIndexesAfter" : 2,
 "ok" : 1,
 "operationTime" : Timestamp(1761149251, 1)
}
```

La versione 2.0 di Planner spiega l'output del piano mostrando la condizione dell'indice e il filtro:

```
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)
}
```

La versione 1.0 di Planner spiega l'output del piano:

```
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)
}
```

La versione 2.0 di Planner colma il divario comportamentale con MongoDB

Ci sono alcune aree in cui la versione 2.0 di Planner colma le lacune comportamentali di MongoDB:

- La versione 2.0 di Planner consente la ricerca dell'indice numerico su array appiattiti per: `$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}}})
```

- Mentre la versione 1.0 di Planner escludeva le stringhe nella proiezione, il comportamento di Planner versione 2.0 si allinea a MongoDB e le considera come valori letterali»

```
// 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 }
```

- La versione 2.0 di Planner, come MongoDB, non consente la proiezione sugli stessi campi «x» e «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 ] }
```

- La versione 2.0 di Planner, come MongoDB, consente la proiezione su documenti secondari:

```
// 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"
}
```

- Con la versione 2.0 di Planner, come MongoDB, la proiezione non supporta i campi successivi all'operatore: $

```
// 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 } ] }
```

- La versione 2.0 di Planner, come MongoDB, consente l'utilizzo di: `$hint`

```
// 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 }
```

Pianificatore di query v3

La versione 3 di Planner in Amazon DocumentDB 8.0 supporta 21 fasi di aggregazione, incluse 6 nuove fasi. Planner V3 è dotato di supporto integrato per comandi distinti. Offre un miglioramento complessivo delle prestazioni fino a 2 volte rispetto a Planner v2 in Amazon DocumentDB 5.0. Tutte le nuove funzionalità e gli operatori di Amazon DocumentDB 8.0 sono compatibili con Planner v3. Le nuove fasi di aggregazione in Amazon DocumentDB 8.0 supportate da Planner v3 includono $replaceWith, $vectorSearch, $merge, $set, $unset, $bucket. Planner v3 supporta anche nuove funzionalità e operatori in Amazon DocumentDB 8.0, tra cui Collation, Views, $merge, $pow, $rand, $DateTrunc, $date, $date. ToParts FromParts

Argomenti

- Prerequisiti
- Selezione della versione 3.0 di Planner come pianificatore di query predefinito
- Best practice
- Limitazioni
- Miglioramenti agli operatori aggregati e distinti
- Pianifica i filtri della cache
- Potenziali differenze di comportamento tra le versioni 1.0, 3.0 e MongoDB di Planner

Prerequisiti

I seguenti prerequisiti si applicano alla versione 3.0 di Planner:

- La versione 3.0 di Planner è disponibile Regioni AWS ovunque sia disponibile la versione 8.0 del motore.
- La versione 3.0 di Planner è il pianificatore di query predefinito quando è selezionata la versione 8.0 del motore.

Selezione della versione 3.0 di Planner come pianificatore di query predefinito

Se hai modificato il pianificatore di query predefinito in Amazon DocumentDB 8.0 e devi tornare al planner v3, puoi farlo dalla console o dalla CLI:

- Segui i passaggi indicati per modificare il gruppo di parametri del Modifica dei parametri del cluster Amazon DocumentDB cluster.
- Per il parametro denominato 'plannerVersion', modifica il valore in 3.0, indicando la versione 3.0 di planner.
- Seleziona Applica immediatamente (selezionando Applica al riavvio la selezione sarà inefficace fino al successivo riavvio del cluster).

Best practice

Per ottenere i risultati previsti, utilizza le seguenti best practice quando applichi la versione 3.0 di Planner:

- In un cluster globale, seleziona lo stesso `plannerVersion` valore (1.0 o 2.0 o 3.0) nei gruppi di parametri del cluster per entrambe le regioni. Tieni presente che la selezione di versioni diverse del planner nelle regioni primarie e secondarie può causare comportamenti e prestazioni delle query incoerenti.
- L'aggiornamento alla versione 3.0 di Planner durante una finestra di manutenzione programmata o durante i periodi di traffico ridotto sarà il meno problematico, poiché potrebbe verificarsi un aumento dei tassi di errore se la versione del pianificatore viene modificata quando i carichi di lavoro sono in esecuzione attiva.

Limitazioni

Le seguenti limitazioni si applicano alla versione 3.0 di Planner:

- La versione 3.0 di Planner non è supportata nei cluster elastici, che torneranno alla versione 1.0 di Planner.

- Mentre Planner v1 consente l'uso di «PlanHint» per garantire che un piano di query specifico venga selezionato dall'ottimizzatore di query, Planner v3 non consente l'uso di «PlanHint» e si basa su ottimizzazioni interne per scegliere il piano migliore per la determinata query.

Miglioramenti agli operatori aggregati e distinti

La versione 3.0 di Planner introduce miglioramenti nelle fasi $aggregate e nel comando $distinct. Di seguito sono riportati alcuni dei miglioramenti più degni di nota.

- Quando possibile, il planner sposta $match nelle fasi precedenti della pipeline, riducendo il numero di documenti elaborati nelle fasi successive.

```
//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 } }
])
```

- Il planner combina automaticamente le fasi $lookup e $unwind quando operano sullo stesso campo, riducendo l'elaborazione intermedia dei dati e migliorando le prestazioni.

```
                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
```

- La versione 3.0 di Planner introduce una nuova strategia di esecuzione di Distinct Scan che migliora significativamente le prestazioni per operazioni distinte su indici a bassa cardinalità.

```
                 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"
                        }
                }
        }
}
```

Pianifica i filtri della cache

La versione 3.0 di Planner supporta i filtri della cache del piano, chiamati anche filtri di indice, che consentono di limitare il set di indici che il planner considera per una specifica forma di query. I filtri vengono impostati con un comando del database e applicati sul server, quindi se si verifica una regressione delle query è possibile attenuarla senza modificare il codice dell'applicazione.

La versione 3.0 di Planner applica i filtri della cache del piano a più comandi rispetto alla versione 2.0 di Planner:

- Il `distinct` comando richiede la versione 3.0 di Planner.
- Il `aggregate` comando richiede la versione 3.0 del planner e Amazon DocumentDB 8.0.2. Un filtro si applica quando la pipeline legge la raccolta attraverso una fase principale. `$match`
- Un filtro impostato sulla raccolta esterna di una `$lookup` o `$graphLookup` fase determina l'indice utilizzato per scansionare quella raccolta esterna. Ciò richiede anche la versione 3.0 di Planner e Amazon DocumentDB 8.0.2.

Per ulteriori informazioni sul riferimento ai comandi, sulle forme di query supportate, sugli esempi funzionanti e su come i filtri interagiscono con i suggerimenti, consulta. Pianifica i filtri della cache

Potenziali differenze di comportamento tra le versioni 1.0, 3.0 e MongoDB di Planner

In alcuni casi limite, è possibile che la versione 3.0 di Planner produca risultati leggermente diversi dalla versione 1.0 di Planner. Questa sezione illustra alcuni esempi di queste possibilità.

Feature Differences

- In una raccolta vuota o quando le fasi precedenti, ad esempio Match, filtrano tutti i documenti, Planner v1 restituisce il «campo» di output :0. Planner v3 e MongoDB non forniranno alcun output.
- Con Plannerv1, {"$skip» :n} non salta se ci sono meno di n documenti restituiti dalla fase precedente. Plannerv3 e MongoDB saltano correttamente indipendentemente dal numero di documenti restituiti.
- Quando una raccolta esterna a cui si fa riferimento in $lookup non esiste, plannerv1 genera un errore. Planner v3 e MongoDB trattano la raccolta esterna come una raccolta vuota ed eseguono $lookup.

```
db.coll.aggregate([
    {$lookup: {from: "does_not_exist", localField: "a", foreignField: "a", as:
 "c"}}
])
```

- Solo Planner v1 consentirà l'utilizzo di più $search in una pipeline Planner v2/v3 genererà un errore e MongoDB non lo supporta.

```
VectorSearch = { "$search": { "vectorSearch": {
   "vector": [0.2, 0.5, 0.8],
   "path": "vectorEmbedding",
   "similarity": "cosine",
   "k": 2,
   "efSearch": 1
   }}}
db.coll.aggregate([VectorSearch, VectorSearch])
```

- Solo Planner v3 funziona quando la fase VectorSearch non è la prima fase. In questo caso, MongoDB genererà un errore e Planner v1 non supporta la fase $vectorSearch.

```
VectorSearch = { {"$vectorSearch": {
    "queryVector": [0.2, 0.5, 0.8],
    "path": "vectorEmbedding",
    "similarity": "euclidean",
    "limit": 4,
    "numCandidates": 100} }

db.coll.aggregate([$match:{}, VectorSearch])
```

Interrogazione di dati geospaziali con Amazon DocumentDB

Questa sezione spiega come interrogare i dati geospaziali con Amazon DocumentDB. Dopo aver letto questa sezione, sarai in grado di rispondere a come archiviare, interrogare e indicizzare i dati geospaziali in Amazon DocumentDB.

Argomenti

- Panoramica di
- Indicizzazione e archiviazione di dati geospaziali
- Esecuzione di query su dati geospaziali
- Limitazioni

Panoramica di

I casi d'uso più comuni di Geospatial riguardano l'analisi di prossimità dei tuoi dati. Ad esempio, «trovare tutti gli aeroporti entro 50 miglia da Seattle» o «trovare i ristoranti più vicini a una determinata località». Amazon DocumentDB utilizza la specifica GeoJSON per rappresentare i dati geospaziali. GeoJSON è una specifica open source per la creazione di forme in uno spazio di coordinate. JSON-formatting Le coordinate GeoJSON catturano sia la longitudine che la latitudine, rappresentando le posizioni su una sfera simile alla terra.

Indicizzazione e archiviazione di dati geospaziali

Amazon DocumentDB utilizza il tipo GeoJSON «Point» per archiviare dati geospaziali. Ogni documento GeoJSON (o documento secondario) è generalmente composto da due campi:

- tipo: la forma rappresentata, che indica ad Amazon DocumentDB come interpretare il campo «coordinate». Al momento, Amazon DocumentDB supporta solo punti

- coordinate, una coppia di latitudine e longitudine rappresentata come oggetto in un array, [longitudine, latitudine]

Amazon DocumentDB utilizza anche indici 2dsphere per indicizzare i dati geospaziali. Amazon DocumentDB supporta i punti di indicizzazione. Amazon DocumentDB supporta l'interrogazione di prossimità con indicizzazione 2dsphere.

Consideriamo uno scenario in cui stai creando un'applicazione per il servizio di consegna di cibo. Vuoi memorizzare le coppie di latitudine e longitudine dei ristoranti in Amazon DocumentDB. Per farlo, crea prima un indice nel campo geospaziale che contenga la coppia di latitudine e longitudine.

```
use restaurantsdb
db.usarestaurants.createIndex({location:"2dsphere"})
```

L'output di questo comando sarebbe simile a questo:

```
{
 "createdCollectionAutomatically" : true,
 "numIndexesBefore" : 1,
 "numIndexesAfter" : 2,
 "ok" : 1
}
```

Dopo aver creato un indice, puoi iniziare a inserire dati nella tua raccolta Amazon DocumentDB.

```
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
      ]
   }
});
```

Esecuzione di query su dati geospaziali

Amazon DocumentDB supporta l'interrogazione di prossimità, inclusione e intersezione di dati geospaziali. Un buon esempio di ricerca di prossimità è la ricerca di tutti i punti (tutti gli aeroporti) che si trovano a meno di una certa distanza e più di una distanza da un altro punto (città). Un buon esempio di ricerca di inclusione consiste nel trovare tutti i punti (tutti gli aeroporti) che si trovano in uno specifico area/polygon (stato di New York). Un buon esempio di ricerca di intersezione è la ricerca di un poligono (stato) che si interseca con un punto (città). Puoi utilizzare i seguenti operatori geospaziali per ottenere informazioni dettagliate dai tuoi dati.

- **`$nearSphere`**- `$nearSphere` è un operatore di ricerca che supporta la ricerca di punti dal più vicino al più lontano da un punto GeoJSON.
- **`$geoNear`**- `$geoNear` è un operatore di aggregazione che supporta il calcolo della distanza in metri da un punto GeoJSON.

- **`$minDistance`**- `$minDistance` è un operatore di ricerca che viene utilizzato insieme `$nearSphere` o `$geoNear` per filtrare documenti che si trovano almeno alla distanza minima specificata dal punto centrale.
- **`$maxDistance`**- `$maxDistance` è un operatore di ricerca utilizzato insieme `$nearSphere` o `$geoNear` per filtrare documenti che si trovano al massimo alla distanza massima specificata dal punto centrale.
- **`$geoWithin`**- `$geoWithin` è un operatore di ricerca che supporta la ricerca di documenti con dati geospaziali che esistono interamente all'interno di una forma specificata, ad esempio un poligono.
- **`$geoIntersects`**- `$geoIntersects` è un operatore di ricerca che supporta la ricerca di documenti i cui dati geospaziali si intersecano con un oggetto GeoJSON specificato.

> **Note**
>
> `$geoNeare` `$nearSphere` richiedono un indice 2dsphere nel campo GeoJSON da utilizzare nella query di prossimità.

Esempio 1

In questo esempio, imparerai come trovare tutti i ristoranti (punti) ordinati in base alla distanza più vicina da un indirizzo (punto).

Per eseguire tale interrogazione, è possibile utilizzare `$geoNear` per calcolare la distanza di un insieme di punti da un altro punto. Puoi anche aggiungerlo `distanceMultiplier` per misurare la distanza in chilometri.

```
db.usarestaurants.aggregate([
   {
      "$geoNear":{
         "near":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "spherical":true,
         "distanceField":"DistanceKilometers",
         "distanceMultiplier":0.001
```

```
      }
   }
])
```

Il comando precedente restituirebbe i ristoranti ordinati per distanza (dal più vicino al più lontano) dal punto specificato. L'output di questo comando sarebbe simile a questo

```
{ "_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**
>
> numLe opzioni `limit` e per lo `$geoNear` stage, così come il limite predefinito di 100 documenti, sono obsolete in Amazon DocumentDB 8.0 (patch del motore versione 4.0.5748) e versioni successive. I seguenti esempi utilizzano `limit` e `num` si applicano solo ad Amazon DocumentDB 8.0 (Engine patch versione 4.0.4448) e versioni precedenti.

Per limitare il numero di risultati in una query, utilizza l'opzione or. `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
      }
   }
])
```

Esempio 2

In questo esempio, imparerai come trovare tutti i ristoranti (punti) nel raggio di 2 chilometri da un indirizzo specifico (punto). Per eseguire tale interrogazione, è possibile utilizzare un valore minimo `$minDistance` e massimo `$nearSphere` compreso `$maxDistance` tra un punto GeoJSON

```
db.usarestaurants.find({
   "location":{
      "$nearSphere":{
         "$geometry":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "$minDistance":1,
```

```
         "$maxDistance":2000
      }
   }
},
{
   "name":1
})
```

Il comando precedente restituirebbe ristoranti a una distanza massima di 2 chilometri dal punto specificato. L'output di questo comando sarebbe simile a questo

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
```

Limitazioni

Amazon DocumentDB non supporta l'interrogazione o l'indicizzazione di Polygons,,, e LineString. MultiPoint MultiPolygon MultiLineString GeometryCollection

Indice parziale

Un indice parziale indicizza i documenti di una raccolta che soddisfa un criterio di filtro specificato. La funzionalità di indice parziale è supportata nei cluster basati su istanze di Amazon DocumentDB 5.0.

Argomenti

- Crea un indice parziale
- Operatori supportati
- Interrogazione utilizzando un indice parziale
- Funzionalità di indicizzazione parziale
- Limitazioni parziali dell'indice

Crea un indice parziale

Per creare un indice parziale, utilizzate il `createIndex()` metodo con l'`partialFilterExpression`opzione. Ad esempio, l'operazione seguente crea un indice composto univoco nella raccolta orders che indicizza i documenti il cui `isDelivered` campo è impostato su true: `OrderID`

```
db.orders.createIndex(
```

```
  {"category": 1, "CustomerId": 1, "OrderId": 1},
  {"unique": true, "partialFilterExpression":
    {"$and": [
      {"OrderId": {"$exists": true}},
      {"isDelivered": {"$eq": false}}
    ]}
  }
)
```

Operatori supportati

- $eq
- $exists
- $and (solo al livello superiore)
- $gt/$gte/$lt/$lte (la scansione dell'indice viene utilizzata solo quando il filtro, previsto nella query, corrisponde esattamente all'espressione parziale del filtro) (vedi Limitazioni)

Interrogazione utilizzando un indice parziale

I seguenti modelli di interrogazione sono possibili utilizzando indici parziali:

- Il predicato della query corrisponde esattamente all'espressione del filtro dell'indice parziale:

```
db.orders.find({"$and": [
    {"OrderId": {"$exists": true}},
    {"isDelivered": {"$eq": false}}
  ]}).explain()
```

- Il risultato previsto dal filtro di interrogazione è un sottoinsieme logico del filtro parziale:

```
db.orders.find({"$and": [
    {"OrderId": {"$exists": true}},
    {"isDelivered": {"$eq": false}},
    {"OrderAmount": {"$eq": "5"}}
  ]}).explain()
```

- Un sottopredicato della query può essere utilizzato insieme ad altri indici:

```
db.orders.createIndex({"anotherIndex":1})
```

```
db.orders.find({ "$or": [
      {"$and": [
        {"OrderId": {"$exists": true}},
        {"isDelivered": {"$eq": false}}
      ]},
      {"anotherIndex": {"$eq": 5}}
    ]
  }).explain()
```

> **Note**
>
> Un pianificatore di query può scegliere di utilizzare una scansione della raccolta anziché una scansione dell'indice se è efficiente farlo. Questo si verifica in genere per raccolte molto piccole o per query che restituirebbero una grande parte di una raccolta.

Funzionalità di indicizzazione parziale

Elenca gli indici parziali

Elenca gli indici parziali con partial FilterExpression utilizzando l'operazione. `getIndex` Ad esempio, l'`getIndex`operazione emessa in elenca gli indici parziali con i campi key, name e partialFilterExpressions:

```
db.orders.getIndexes()
```

Questo esempio restituisce il seguente risultato:

```
[
  {
    "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}}
      ]
    }
  }
]
```

Espressione di filtro parziale multipla sulla stessa chiave:order

È possibile creare diversi indici parziali per le stesse combinazioni di campi (key:order). Questi indici devono avere un nome diverso.

```
db.orders.createIndex(
  {"OrderId":1},
  {
    name:"firstPartialIndex",
    partialFilterExpression:{"OrderId":{"$exists": true}}
  }
)
```

```
db.orders.createIndex(
  {"OrderId":1},
  {
    name:"secondPartialIndex",
    partialFilterExpression:{"OrderId":{"$gt": 1000}}
  }
)
```

Esegui `getIndexes` l'operazione per elencare tutti gli indici della raccolta:

```
db.orders.getIndexes()
```

Questi esempi restituiscono il seguente risultato:

```
[
  {
    "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**
>
> I nomi degli indici devono essere diversi e devono essere eliminati solo per nome.

Indici con proprietà parziali e TTL

È inoltre possibile creare indici con proprietà parziali e TTL specificando entrambe le opzioni durante la creazione dell'indice. `partialFilterExpression` `expireAfterSeconds` Ciò consente di fornire un maggiore controllo su quali documenti vengono ora rimossi da una raccolta.

Ad esempio, potresti avere un indice TTL che identifica i documenti da eliminare dopo un certo periodo di tempo. Ora puoi fornire condizioni aggiuntive su quando eliminare i documenti utilizzando l'opzione di indice parziale:

```
db.orders.createIndex(
    { "OrderTimestamp": 1 },
    {
        expireAfterSeconds: 3600 ,
        partialFilterExpression: { "isDelivered": { $eq: true } }
    }
)
```

Questo esempio restituisce il seguente risultato:

```
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1,
        "operationTime" : Timestamp(1234567890, 1)
}
```

Eseguite l'`getIndexes`operazione per elencare gli indici presenti nella raccolta:

```
db.orders.getIndexes()
[
    {
        "v" : 4,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.orders"
    }
```

Questo esempio restituisce il seguente risultato:

```
[
    {
        "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
    }
]
```

Limitazioni parziali dell'indice

Le seguenti limitazioni si applicano alla funzionalità di indice parziale:

- Le query di disuguaglianza in Amazon DocumentDB utilizzeranno un indice parziale solo quando il predicato del filtro di query corrisponde esattamente a ed è `partialFilterExpression` dello stesso tipo di dati.

  > **Note**
  >
  > `$hint`Non può essere utilizzato nemmeno per forzare IXSCAN nel caso precedente.

  Nell'esempio seguente, `partialFilterExpression` viene applicato solo `field1` ma non `field2` a:

```
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
```

- Gli operatori A `partialFilterExpression` con array non sono supportati. La seguente operazione genererà un errore:

```
db.orders.createIndex(
  {"CustomerId":1},
  {'partialFilterExpression': {'OrderId': {'$eq': [1000, 1001, 1002]}}}
)
```

- I seguenti operatori non sono supportati nel FilterExpression campo parziale:
  - `$all`(operatore di matrice)
  - `$mod`(operatore di matrice)
  - `$or`
  - `$xor`
  - `$not`
  - `$nor`
- Il tipo di dati dell'espressione del filtro e del filtro devono essere gli stessi.

Esecuzione di ricerche di testo con Amazon DocumentDB

La funzionalità di ricerca full-text nativa di Amazon DocumentDB (Text Index v1) consente di eseguire ricerche di testo su set di dati testuali di grandi dimensioni utilizzando indici di testo per scopi speciali. Questa sezione descrive le funzionalità della funzionalità di indice di testo e fornisce istruzioni su come creare e utilizzare indici di testo in Amazon DocumentDB. Sono elencate anche le limitazioni alla ricerca testuale.

Argomenti

- Funzionalità supportate
- Utilizzo dell'indice di testo di Amazon DocumentDB
- Differenze con MongoDB
- Buone pratiche e linee guida
- Indice di testo V2
- Limitazioni

Funzionalità supportate

La ricerca di testo di Amazon DocumentDB supporta le seguenti funzionalità compatibili con l'API MongoDB:

- Crea indici di testo su un singolo campo.
- Crea indici di testo composti che includono più di un campo di testo.
- Esegui ricerche con una o più parole.
- Controlla i risultati della ricerca utilizzando i pesi.
- Ordina i risultati della ricerca per punteggio.
- Usa l'indice di testo nella pipeline di aggregazione.
- Cerca una frase esatta.

Utilizzo dell'indice di testo di Amazon DocumentDB

Per creare un indice di testo su un campo contenente dati di tipo stringa, specifica la stringa «testo» come mostrato di seguito:

Indice a campo singolo:

```
db.test.createIndex({"comments": "text"})
```

Questo indice supporta le query di ricerca di testo nel campo stringa «commenti» della raccolta specificata.

Crea un indice di testo composto su più di un campo stringa:

```
db.test.createIndex({"comments": "text", "title":"text"})
```

Questo indice supporta le query di ricerca di testo nei campi di stringa «commenti» e «titolo» della raccolta specificata. È possibile specificare fino a 30 campi quando si crea un indice di testo composto. Una volta create, le query di ricerca testuale interrogheranno tutti i campi indicizzati.

> **Note**
>
> È consentito un solo indice di testo per ogni raccolta.

Inserire un indice di testo in una raccolta Amazon DocumentDB

Puoi utilizzarlo `getIndexes()` nella tua raccolta per identificare e descrivere gli indici, inclusi gli indici di testo, come mostrato nell'esempio seguente:

```
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
   }
]
```

Dopo aver creato un indice, inizia a inserire i dati nella tua raccolta Amazon DocumentDB.

```
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"}])
```

Esecuzione di query di ricerca testuale

Esegui una query di ricerca testuale con una sola parola

Dovrai utilizzare gli `$search` operatori `$text` e per eseguire ricerche di testo. L'esempio seguente restituisce tutti i documenti in cui il campo di testo indicizzato contiene la stringa «apple» o «apple» in altri formati come «apples»:

```
db.test.find({$text: {$search: "apple"}})
```

Output:

L'output di questo comando è simile al seguente:

```
{ "_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" }
```

Esegui una ricerca testuale composta da più parole

Puoi anche eseguire ricerche di testo con più parole sui tuoi dati Amazon DocumentDB. Il comando seguente restituisce i documenti con un campo indicizzato di testo contenente «mela» o «torta»:

```
db.test.find({$text: {$search: "apple pie"}})
```

Output:

L'output di questo comando è simile al seguente:

```
{ "_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" }
```

Esegui una ricerca testuale composta da più parole

Per una ricerca con più parole, usa questo esempio:

```
db.test.find({$text: {$search: "\"apple pie\""}})
```

Output:

Il comando precedente restituisce documenti con un campo indicizzato di testo contenente la frase esatta «torta di mele». L'output di questo comando è simile al seguente:

```
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
```

Esegui una ricerca testuale con filtri

Puoi anche combinare la ricerca testuale con altri operatori di query per filtrare i risultati in base a criteri aggiuntivi:

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "interest"}}]})
```

Output:

Il comando precedente restituisce i documenti con un campo indicizzato di testo contenente qualsiasi forma di «interesse» e un «star_rating» pari a 5. L'output di questo comando è simile al seguente:

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
{ "_id" : 6, "star_rating" : 5, "comments" : "interested in couch for sale, year
 2022" }
```

Limita il numero di documenti restituiti in una ricerca testuale

Puoi scegliere di limitare il numero di documenti restituiti utilizzando`limit`:

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "couch"}}]}).limit(1)
```

Output:

Il comando precedente restituisce un risultato che soddisfa il filtro:

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
```

Ordina i risultati in base al punteggio del testo

L'esempio seguente ordina i risultati della ricerca testuale in base al punteggio del testo:

```
db.test.find({$text: {$search: "apple"}}, {score: {$meta: "textScore"}}).sort({score:
 {$meta: "textScore"}})
```

Output:

Il comando precedente restituisce i documenti con un campo indicizzato di testo contenente «mela» o «mela» in altri formati come «mele» e ordina il risultato in base alla pertinenza del documento rispetto al termine di ricerca. L'output di questo comando è simile al seguente:

```
{ "_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 }
```

`$texte` `$search` sono supportati anche per i `delete` comandi `aggregate` `countfindAndModify`,`update`,, e.

Operatori di aggregazione

Pipeline di aggregazione utilizzando **`$match`**

```
db.test.aggregate(
   [{ $match: { $text: { $search: "apple pie" } } }]
)
```

Output:

Il comando precedente restituisce i seguenti risultati:

```
{ "_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" }
```

Una combinazione di altri operatori di aggregazione

```
db.test.aggregate(
   [
      { $match: { $text: { $search: "apple pie" } } },
      { $sort: { score: { $meta: "textScore" } } },
      { $project: { score: { $meta: "textScore" } } }
   ]
)
```

Output:

Il comando precedente restituisce i seguenti risultati:

```
{ "_id" : 4, "score" : 0.6079270860936958 }
{ "_id" : 1, "score" : 0.3039635430468479 }
{ "_id" : 2, "score" : 0.3039635430468479 }
{ "_id" : 3, "score" : 0.3039635430468479 }
```

Specifica più campi durante la creazione di un indice di testo

Puoi assegnare pesi a un massimo di tre campi nell'indice di testo composto. Il peso predefinito assegnato a un campo in un indice di testo è uno (1). Il peso è un parametro opzionale e deve essere compreso tra 1 e 100000.

```
db.test.createIndex(
   {
     "firstname": "text",
     "lastname": "text",
     ...
   },
   {
     weights: {
       "firstname": 5,
       "lastname":10,
       ...
     },
     name: "name_text_index"
   }
 )
```

Differenze con MongoDB

La funzionalità di indice di testo di Amazon DocumentDB utilizza un indice invertito con un algoritmo di frequenza dei termini. Gli indici di testo sono scarsi per impostazione predefinita. A causa delle differenze nella logica di analisi, nei delimitatori di tokenizzazione e altro, lo stesso set di risultati di MongoDB potrebbe non essere restituito per lo stesso set di dati o la stessa forma di query.

Esistono le seguenti differenze aggiuntive tra l'indice di testo di Amazon DocumentDB e MongoDB:

- Gli indici composti che utilizzano indici non testuali non sono supportati.
- Gli indici di testo di Amazon DocumentDB non fanno distinzione tra maiuscole e minuscole.

- L'indice di testo supporta solo la lingua inglese.
- L'indicizzazione del testo dei campi a matrice (o a più chiavi) non è supportata. Ad esempio, la creazione di un indice di testo su «a «con il documento {«a»: [«apple», «pie»]} avrà esito negativo.
- L'indicizzazione del testo con caratteri jolly non è supportata.
- Gli indici di testo univoci non sono supportati.
- L'esclusione di un termine non è supportata.

Buone pratiche e linee guida

- Per prestazioni ottimali nelle query di ricerca di testo che comportano l'ordinamento per punteggi di testo, crea l'indice di testo prima di caricare i dati.
- Gli indici di testo richiedono spazio di archiviazione aggiuntivo per una copia interna ottimizzata dei dati indicizzati. Ciò comporta ulteriori implicazioni in termini di costi.

Indice di testo V2

Amazon DocumentDB 8.0 introduce una nuova versione di text index (V2) che modifica il parser di ricerca testuale sottostante per aumentare la compatibilità con MongoDB.

Oltre alle funzionalità fornite dagli indici di testo V1, gli indici di testo V2 forniscono anche il seguente supporto:

- Quando possibile, il planner sposta $match nelle fasi precedenti della pipeline, riducendo il numero di documenti elaborati nelle fasi successive.
- Migliore tokenizzazione dei caratteri speciali nei campi di testo, come e-mail, URL e percorsi di file. V2 può analizzare e abbinare singoli token all'interno di queste stringhe, mentre V1 no.

```
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" }
```

Limitazioni

La ricerca testuale presenta le seguenti limitazioni in Amazon DocumentDB:

- Gli indici di testo memorizzano i lessemi e le relative informazioni sulla posizione. La dimensione combinata di tutti i lessemi e delle relative informazioni sulla posizione, all'interno di un singolo documento, è limitata a 1 MB.

Indici

Gli indici migliorano le prestazioni delle query consentendo ad Amazon DocumentDB di individuare rapidamente i documenti senza scansionare tutti i documenti di una raccolta. Puoi creare indici sui campi che l'applicazione interroga frequentemente per ridurre latenza e costi. I/O

Per ulteriori argomenti relativi agli indici, consulta:

- Gestione degli indici Amazon DocumentDB— creazione e creazione di indici
- Utilizzo degli indici— best practice di indicizzazione
- Indici per la risoluzione dei problemi— risoluzione degli errori e dei problemi di compilazione degli indici
- Come posso analizzare l'utilizzo degli indici e identificare gli indici inutilizzati?— ricerca di indici inutilizzati o mancanti

Argomenti

- gestione degli indici
- Tipi di indice
- Proprietà dell'indice

gestione degli indici

Questa sezione tratta argomenti sulla gestione degli indici per Amazon DocumentDB.

Argomenti

- Creazione dell'indice
- Manutenzione degli indici
- Procedure consigliate per lavorare con gli indici

Creazione dell'indice

La creazione di indici in Amazon DocumentDB richiede una serie di decisioni:

- Quanto velocemente deve essere completato?
- La raccolta può essere inaccessibile durante la compilazione?
- Quanta potenza di calcolo di un'istanza può essere allocata alla build?
- Che tipo di indice deve essere creato?

Questa sezione ti aiuta a rispondere a queste domande e fornisce i comandi e gli esempi di monitoraggio per creare un indice Amazon DocumentDB sulla tua raccolta di cluster basata su istanze.

Linee guida

Le seguenti linee guida includono limiti di base e compromessi di configurazione nella creazione di nuovi indici:

- Supporto delle versioni di Amazon DocumentDB: mentre l'indicizzazione di un singolo lavoratore è supportata in tutte le versioni di Amazon DocumentDB, l'indicizzazione di più lavoratori è supportata solo nelle versioni 4.0 e 5.0 di Amazon DocumentDB.
- Compromesso in termini di prestazioni: l'aumento del numero di lavoratori nel processo di creazione dell'indice aumenta l'utilizzo della CPU e l'IO di lettura sull'istanza primaria del database Amazon DocumentDB. Le risorse necessarie per creare un nuovo indice non saranno disponibili per il carico di lavoro in esecuzione.
- Cluster elastici: l'indicizzazione parallela non è supportata sui cluster elastici Amazon DocumentDB.

- Cluster serverless: l'indicizzazione parallela su cluster serverless richiede una capacità minima di almeno 32 DCU. Al di sotto di questa soglia, il `workers` parametro è accettato ma la creazione dell'indice viene eseguita con un singolo worker. Il numero massimo di lavoratori paralleli varia in base alla capacità minima configurata: sono disponibili circa `min_DCU / 8` lavoratori. Ad esempio, con 32 DCU è possibile utilizzare fino a 4 lavoratori e con 64 DCU è possibile utilizzare fino a 8 lavoratori.

  > **Note**
  >
  > È necessario riavviare il cluster dopo aver modificato l'impostazione della capacità minima per rendere effettivo il nuovo limite di lavoratori paralleli.

- Numero massimo di lavoratori (basato sull'istanza): il numero massimo di lavoratori che è possibile configurare dipende dalle dimensioni dell'istanza principale nel cluster di database. È la metà del numero totale di vCPU sull'istanza primaria del cluster di database. Ad esempio, è possibile eseguire un massimo di 32 worker su un'istanza db.r6g.16xlarge con 64 vCPU.

> **Note**
>
> La creazione parallela dell'indice è supportata sulle istanze 2xlarge e successive.

- Numero minimo di lavoratori: il numero minimo di lavoratori che puoi configurare è uno. L'impostazione predefinita per la creazione dell'indice su cluster basati su istanze è di due worker. Tuttavia, è possibile ridurre il numero di lavoratori a uno utilizzando l'opzione «worker threads». Questo eseguirà il processo con un solo lavoratore.
- Compressione dell'indice: Amazon DocumentDB non supporta la compressione dell'indice. Le dimensioni dei dati per gli indici potrebbero essere maggiori rispetto a quando si utilizzano altre opzioni.
- Indicizzazione di più raccolte: metà delle vCPU sull'istanza principale del cluster di database può essere utilizzata per i lavoratori configurati che eseguono la creazione di indici su più raccolte.
- Tipi di indice: consulta l'elenco Tipi di indice dei tipi di indice supportati su Amazon DocumentDB.

Tipi di compilazione dell'indice

I quattro tipi di build dell'indice sono:

- Foreground: la build dell'indice in primo piano blocca tutte le altre operazioni del database sulla raccolta in cui l'indice viene creato, fino al completamento della creazione dell'indice. La build in primo piano di Amazon DocumentDB è composta da cinque fasi.
- Foreground (unique): le build con indice in primo piano a documento singolo (univoco) bloccano altre operazioni del database, ad esempio le normali compilazioni in primo piano. A differenza della build di base in primo piano, la build univoca utilizza una fase aggiuntiva (ordinamento delle chiavi 2) per cercare le chiavi duplicate. La build in primo piano (unica) è composta da sei fasi.
- Background: la creazione dell'indice in background consente di eseguire altre operazioni del database in primo piano durante la creazione dell'indice. La build in background di Amazon DocumentDB è composta da otto fasi.
- Sfondo (univoco): le compilazioni di indici in background per un singolo documento (univoco) consentono di eseguire altre operazioni del database in primo piano durante la creazione dell'indice. A differenza della build di base in background, la build univoca utilizza una fase aggiuntiva (ordinamento delle chiavi 2) per cercare le chiavi duplicate. La build di sfondo (unica) è composta da nove fasi.

Il tipo di build dell'indice predefinito è foreground.

Per creare un indice in background, aggiungete il parametro «background»: true durante la creazione dell'indice.

Nozioni di base

Per iniziare la creazione dell'indice su una raccolta, utilizzate il comando createIndexes. Per impostazione predefinita, il comando eseguirà due worker paralleli che aumentano la velocità del processo di creazione dell'indice di due volte.

Ad esempio, il seguente processo di comando dimostra come creare un indice per il campo «user_name» in un documento e aumentare la velocità del processo di indicizzazione a quattro worker:

1. Crea indici in background utilizzando due worker paralleli sul cluster:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "background":true }]})
```

2. Per ottimizzare la velocità del processo di creazione dell'indice, è possibile specificare il numero di lavoratori utilizzando l'opzione «worker threads» («workers»:<number>) nel comando db.runCommand createIndexes.

   Aumentate la velocità del processo fino a quattro lavoratori paralleli:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "workers":4}]})
```

> **Note**
>
> Maggiore è il numero di lavoratori, più velocemente procede la creazione dell'indice. Tuttavia, maggiore è il numero di lavoratori, maggiore è il carico sulle vCPU e sull'IO di lettura dell'istanza primaria. Assicurati che il cluster sia sufficientemente predisposto per gestire l'aumento del carico senza compromettere gli altri carichi di lavoro.

Stato di avanzamento dell'indicizzazione

Il processo di creazione dell'indice funziona inizializzando, analizzando le raccolte, ordinando le chiavi e, infine, inserendo le chiavi tramite un generatore di indici. Il processo prevede fino a sei fasi quando viene eseguito in primo piano e fino a nove fasi quando viene eseguito in background. È possibile visualizzare le metriche sullo stato, ad esempio la percentuale di completamento, il numero totale di blocchi di archiviazione scansionati, le chiavi ordinate e le chiavi inserite, fase per fase.

Monitora l'avanzamento del processo di indicizzazione utilizzando il comando db.currentOp () nella shell mongo. Il completamento al 100% dell'ultima fase mostra che tutti gli indici sono stati creati con successo:

```
db.currentOp({"command.createIndexes": { $exists : true } })
```

> **Note**
>
> La visualizzazione dello stato di avanzamento dell'indicizzazione è supportata solo su Amazon DocumentDB 5.0.

Fasi di creazione dell'indice

Fase	Primo piano	Primo piano (unico)	Contesto	Sfondo (unico)
Inizializzazione	1	1	1	1
indice di costruzione: inizializzazione	2	2	2	2
indice di costruzione: raccolta di scansioni	3	3	3	3
indice degli edifici: chiavi di ordinamento 1	4	4	4	4
indice di costruzio ne: chiavi di ordinamento 2		5		5
indice di costruzione: inserimento di chiavi	5	6	5	6
convalida: indice di scansione			6	7

Fase	Primo piano	Primo piano (unico)	Contesto	Sfondo (unico)
convalida: ordinamento delle tuple			7	8
convalida: scansione della raccolta			8	9

- inizializzazione: createIndex sta preparando il generatore di indici. Questa fase dovrebbe essere molto breve.
- building index: inizializzazione - Il generatore di indici si sta preparando a creare l'indice. Questa fase dovrebbe essere molto breve.
- building index: scanning collection - Il generatore di indici sta eseguendo una scansione della raccolta per raccogliere le chiavi dell'indice. L'unità di misura è «blocchi».

> **Note**
>
> Se più di un lavoratore è configurato per la creazione dell'indice, questo viene visualizzato in questa fase. La fase di «scansione della raccolta» è l'unica fase che utilizza più worker durante il processo di creazione dell'indice. Tutte le altre fasi mostreranno un singolo lavoratore.

- building index: ordinamento delle chiavi 1 - Il generatore di indici sta ordinando le chiavi di indice raccolte. L'unità di misura è «chiavi».
- building index: sorting keys 2 - Il generatore di indici sta ordinando le chiavi di indice raccolte che corrispondono a tuple morte. Questa fase esiste solo per la creazione di un indice univoco. L'unità di misura è «chiavi».
- building index: inserimento di chiavi - Il generatore di indici sta inserendo le chiavi dell'indice nel nuovo indice. L'unità di misura è «chiavi».
- convalida: indice di scansione - CreateIndex sta scansionando l'indice per trovare le chiavi che devono essere convalidate. L'unità di misura è «blocchi».

- validazione: ordinamento delle tuple - CreateIndex sta ordinando l'output della fase di scansione dell'indice.
- convalida: scansione della raccolta - CreateIndex sta scansionando la raccolta per convalidare le chiavi di indice trovate nelle due fasi precedenti. L'unità di misura è «blocchi».

Esempio di output per la creazione dell'indice

Nel seguente esempio di output (creazione dell'indice in primo piano), viene mostrato lo stato della creazione dell'indice. Il campo «msg» riassume l'avanzamento della compilazione indicando la fase e la percentuale di completamento della compilazione. Il campo «workers» indica il numero di lavoratori utilizzati durante quella fase della creazione dell'indice. Il campo «avanzamento» mostra i numeri effettivi utilizzati per calcolare la percentuale di completamento.

> **Note**
>
> I campi IndexBuildName «current», «msg» e «progress» non sono supportati in Amazon DocumentDB versione 4.0.

```
{
    "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
}
```

Risorse per la risoluzione dei problemi

Consulta le seguenti risorse sulla risoluzione dei problemi di creazione degli indici:

- La creazione dell'indice non riesce
- Problemi di latenza nella creazione dell'indice in background e non riesce

Manutenzione degli indici

Index bloat

Amazon DocumentDB utilizza Multi-Version Concurrency Control (MVCC) per gestire transazioni simultanee. Quando i documenti vengono eliminati o aggiornati, le versioni precedenti rimangono nelle raccolte e negli indici come versioni «morte». Il processo di garbage collection recupera automaticamente lo spazio da queste versioni obsolete per operazioni future.

L'ingrandimento dell'indice si verifica quando gli indici di una raccolta diventano più grandi a causa dell'accumulo di voci di indice morte o obsolete o della frammentazione all'interno delle pagine. La percentuale riportata rappresenta la quantità di spazio nell'indice che può essere utilizzata dalle voci future dell'indice. Questo ingrossamento consuma spazio sia nella cache del buffer che nella memoria. Se desideri rimuovere il gonfiore, dovrai ricostruire gli indici.

Esempio: esegui il comando seguente per determinare lo spazio di archiviazione inutilizzato per il tuo indice:

```
db.coll.aggregate({$indexStats:{}});
```

Il risultato è simile a questo:

```
{
    "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
    }
}
```

È possibile ricostruire gli indici senza tempi di inattività utilizzando il comando ReIndex, che richiede una scansione dell'intera raccolta. Vedi Manutenzione dell'indice utilizzando reIndex.

Manutenzione dell'indice tramite reIndex

ReIndex è un comando utilizzato per ricostruire un indice. Viene in genere utilizzato quando un indice è danneggiato o inefficiente. Nel tempo, gli indici possono accumulare spazio inutilizzato a causa di numerosi aggiornamenti, inserimenti o eliminazioni, con conseguente riduzione delle prestazioni. La reindicizzazione aiuta a rimuovere lo spazio inutilizzato e a ripristinare l'efficienza dell'indice.

Linee guida ReIndex

- ReIndex è supportato solo su Amazon DocumentDB 5.0 e versioni successive
- ReIndex viene sempre eseguito in background.

- Amazon DocumentDB supporta la reindicizzazione di un singolo indice in background, consentendo l'utilizzo di più lavoratori. Il vecchio indice è utilizzabile tramite query quando il processo reIndex è in esecuzione.
- Amazon DocumentDB supporta l'indicizzazione del rapporto sullo stato di avanzamento tramite CurrentOp. Puoi visualizzare fasi di creazione dell'indice simili alle fasi di creazione dell'indice visualizzate durante la creazione dell'indice. L'unica differenza è che reIndex ha sempre otto fasi, indipendentemente dal fatto che sia unico o meno. Non esiste una fase «building index: sorting keys 2".
- ReIndex può essere eseguito contemporaneamente a qualsiasi comando ad eccezione dei comandi relativi all'indice sulla stessa raccolta: createIndexes, DropIndexes, CollMod e RenameCollection.
- ReIndex non è attualmente supportato per gli indici di testo, geospaziali e vettoriali.
- Il supporto reIndex per gli indici parziali è disponibile nella versione del motore 8.0.2 e successive.

compilazione ReIndex

Usa il seguente comando per ricostruire il tuo indice:

```
db.runCommand({ reIndex: "collection-name", index: "index-name"})
```

Facoltativamente, puoi anche controllare il numero di lavoratori assegnati al processo di ricostruzione:

```
db.runCommand({ reIndex: "collection-name", index: "index-name", workers: number })
```

Per informazioni specifiche sulla gestione degli indici con Java, vedere. Gestione degli indici in Amazon DocumentDB con Java

Procedure consigliate per lavorare con gli indici

Creazione di indici

Quando importi dati in Amazon DocumentDB, devi creare i tuoi indici prima di importare set di dati di grandi dimensioni. Puoi utilizzare Amazon DocumentDB Index Tool per estrarre gli indici

da un'istanza MongoDB o da una directory mongodump in esecuzione e creare tali indici in un cluster Amazon DocumentDB. Per ulteriori informazioni sulle migrazioni, consulta Migrazione e aggiornamento.

Selettività degli indici

Limita la creazione dell'indice ai campi in cui il numero di valori duplicati è inferiore all'1% del numero totale di documenti della raccolta. Ad esempio, se la raccolta contiene 100.000 documenti, crea indici solo nei campi in cui lo stesso valore è presente 1.000 volte o meno.

La scelta di un indice con un numero elevato di valori univoci (ovvero un'elevata cardinalità) garantisce che le operazioni di filtro restituiscano un numero limitato di documenti, garantendo quindi buone prestazioni durante le scansioni degli indici. Un esempio di indice ad elevata cardinalità è un indice univoco, che garantisce che i predicati di uguaglianza restituiscano al massimo un singolo documento. Esempi di bassa cardinalità includono un indice su un campo booleano e un indice nel giorno della settimana. A causa delle loro scarse prestazioni, è improbabile che l'ottimizzatore di query del database scelga indici a bassa cardinalità. Allo stesso tempo, gli indici a bassa cardinalità continuano a consumare risorse come spazio su disco e. I/Os Come regola generale, occorre indicare indici su campi in cui la frequenza tipica del valore è l'1% della dimensione totale della raccolta.

Inoltre, è consigliabile creare solo indici su campi comunemente utilizzati come un filtro e cercare regolarmente indici inutilizzati. Per ulteriori informazioni, consulta the section called “Analizza l'utilizzo degli indici e identifica gli indici non utilizzati”.

Impatto degli indici sulla scrittura dei dati

Sebbene gli indici possano migliorare le prestazioni delle query evitando la necessità di eseguire la scansione di tutti i documenti di una raccolta, questo miglioramento comporta un compromesso. Per ogni indice di una raccolta, ogni volta che un documento viene inserito, aggiornato o eliminato, il database deve aggiornare la raccolta e scrivere i campi in ciascuno degli indici per la raccolta. Ad esempio, se una raccolta dispone di nove indici, il database deve eseguire dieci scritture prima di riconoscere l'operazione al client. Pertanto, ogni indice aggiuntivo comporta una latenza di scrittura aggiuntiva e un aumento dello I/Os spazio di archiviazione complessivo utilizzato.

Le istanze del cluster devono essere dimensionate in modo appropriato per mantenere tutta la memoria del set di lavoro. Ciò evita la necessità di leggere continuamente le pagine di indice dal volume di archiviazione, il che influisce negativamente sulle prestazioni e genera costi più elevati. I/O Per ulteriori informazioni, consulta Dimensionamento delle istanze.

Per ottenere prestazioni ottimali, ridurre al minimo il numero di indici nelle raccolte, aggiungendo solo gli indici necessari per migliorare le prestazioni per le query comuni. Mentre i carichi di lavoro variano, una buona linea guida consiste nel mantenere il numero di indici per raccolta a cinque o meno.

Identificazione degli indici mancanti

È consigliabile identificare regolarmente gli indici mancanti. Per ulteriori informazioni, consulta the section called “Identifica gli indici mancanti”.

Identificazione degli indici non utilizzati

Come best practice, è consigliabile identificare e rimuovere regolarmente gli indici inutilizzati. Per ulteriori informazioni, consulta the section called “Analizza l'utilizzo degli indici e identifica gli indici non utilizzati”.

Tipi di indice

Argomenti

- Indici a campo singolo
- Indici composti
- Multi-key indici
- Indici di testo
- Indici geospaziali
- Indici vettoriali

Indici a campo singolo

Gli indici a campo singolo memorizzano le informazioni di ogni singolo campo in una raccolta di documenti. Per impostazione predefinita, tutte le raccolte hanno un indice sul `_id` campo. È possibile aggiungere indici aggiuntivi per velocizzare le query e le operazioni importanti.

Gli indici a campo singolo sono utili quando:

- L'applicazione interroga o filtra frequentemente i documenti in base ai valori di un campo specifico.
- È necessario ordinare in modo efficiente i documenti in base a un campo particolare.
- Vuoi garantire l'unicità di un campo specifico tra i documenti di una raccolta creando un indice univoco.

Proprietà dell'indice supportate

Opzione	3.6	4.0	5.0	8.0	Cluster elastico
name	Sì	Sì	Sì	Sì	Sì
unico	Sì	Sì	Sì	Sì	Sì
sparso *	Sì	Sì	Sì	Sì	Sì
parziale * FilterExpression	No	No	Sì	Sì	No
scadonoAfterSeconds	Sì	Sì	Sì	Sì	Sì

* Le `partialFilterExpression` opzioni `sparse` and non possono essere utilizzate insieme nella stessa definizione di indice. Se si tenta di creare un indice con queste opzioni, l'indice avrà esito negativo e verrà visualizzato il seguente errore:

```
Error in specification: cannot mix partialFilterExpression and sparse options
```

Creazione di un indice a campo singolo

Utilizzate il `createIndex()` metodo per creare un indice di campo singolo. La sintassi è:`db.collection.createIndex(<key>, <options>)`.

Il `key` parametro è un documento JSON che specifica il campo e l'ordinamento dell'indice:

```
{
  "<field>": <1 (ascending)|-1 (descending)>
}
```

Il `options` parametro è un documento JSON che specifica le opzioni per l'indice:

```
{
```

```
  "name": "<name>",
  "unique": <true | false>,
  "sparse": <true | false>,
  "partialFilterExpression": <filter expression>,
  "expireAfterSeconds": <seconds before expiry>
}
```

Vedi Proprietà dell'indice per esempi di creazione di indici a campo singolo.

Indici composti

Gli indici composti memorizzano le informazioni da due o più campi in una raccolta di documenti, consentendo di eseguire interrogazioni sul primo campo o su qualsiasi campo con prefisso. Questi indici ottimizzano le prestazioni per le query che filtrano su più campi contemporaneamente o combinano il filtraggio con le operazioni di ordinamento. Sono efficaci anche per le query a condizione singola sui campi indicizzati più a sinistra. Il database sfrutta queste voci dell'indice per individuare in modo efficiente i documenti corrispondenti senza eseguire scansioni complete della raccolta.

Gli indici di campo composti sono utili quando:

- È necessario filtrare su più campi contemporaneamente.
- È necessario combinare il filtraggio con le operazioni di ordinamento.

Proprietà dell'indice supportate

Opzione	3.6	4.0	5.0	8.0	Cluster elastico
name	Sì	Sì	Sì	Sì	Sì
unico	Sì	Sì	Sì	Sì	Sì
sparso *	Sì	Sì	Sì	Sì	Sì
parziale * FilterExpression	No	No	Sì	Sì	No

* Le `partialFilterExpression` opzioni `sparse` and non possono essere utilizzate insieme nella stessa definizione di indice. Se si tenta di creare un indice con queste opzioni, l'indice avrà esito negativo e verrà visualizzato il seguente errore:

```
Error in specification: cannot mix partialFilterExpression and sparse options
```

Creazione di un indice composto

Utilizzate il `createIndex()` metodo per creare un indice composto. La sintassi del metodo è: `db.collection.createIndex(<keys>, <options>)`

Il `keys` parametro è un documento JSON che specifica i campi e l'ordinamento dell'indice:

```
{
  "<field 1>": <1 (ascending)|-1 (descending)>,
  "<field 2>": <1 (ascending)|-1 (descending)>,
  ...
}
```

Nota che solo un campo può essere un array in un indice composto. Se si tenta di creare un indice composto su due o più campi di matrice, l'operazione avrà esito negativo e verrà visualizzato il seguente errore:

```
multiple fields of compound index cannot be arrays
```

Il `options` parametro è un documento JSON che specifica le opzioni per l'indice:

```
{
  "name": "<name>",
  "unique": <true | false>,
  "sparse": <true | false>,
  "partialFilterExpression": <filter expression>
}
```

Vedi Proprietà dell'indice per esempi di creazione di indici composti.

Multi-key indici

Per i campi con un valore di matrice, un indice a più chiavi consente di creare una chiave di indice per ogni elemento dell'array. L'indicizzazione di un array crea una voce di indice per ogni elemento dell'array.

Multi-key gli indici sono utili quando l'applicazione interroga o filtra frequentemente i documenti in base ai valori negli array.

Proprietà dell'indice supportate

Opzione	3.6	4.0	5.0	8.0	Cluster elastico
name	Sì	Sì	Sì	Sì	Sì
unico	Sì	Sì	Sì	Sì	Sì
sparso *	Sì	Sì	Sì	Sì	Sì
parziale * FilterExpression	No	No	Sì	Sì	No
scadonoAfterSeconds	Sì	Sì	Sì	Sì	Sì

* Le `partialFilterExpression` opzioni `sparse` and non possono essere utilizzate insieme nella stessa definizione di indice. Se si tenta di creare un indice con queste opzioni, l'indice avrà esito negativo e verrà visualizzato il seguente errore:

```
Error in specification: cannot mix partialFilterExpression and sparse options
```

Creazione di un indice a più chiavi

Utilizzate il `createIndex()` metodo per creare un indice a più chiavi. La sintassi del metodo è: `db.collection.createIndex(<key>, <options>)`

Il `key` parametro è un documento JSON che specifica il campo e l'ordinamento dell'indice:

```
{
  "<field>": <1 (ascending)|-1 (descending)>
}
```

Il `options` parametro è un documento JSON che specifica le opzioni per l'indice:

```
{
  "name": "<name>",
  "unique": <true | false>,
  "sparse": <true | false>,
  "partialFilterExpression": <filter expression>,
  "expireAfterSeconds": <seconds before expiry>
}
```

L'esempio seguente crea un indice a più chiavi sul `categories` campo ordinato in ordine crescente con il nome: `book_categories`

```
db.collection.createIndex(
  {
    "categories": 1
  },
  {
    "name": "book_categories"
  }
)
```

Vedi Proprietà dell'indice per esempi di creazione di indici a più chiavi.

Indici di testo

Gli indici di testo sono utili per cercare parole o frasi nei campi di testo all'interno dei documenti, consentendo di eseguire funzionalità di ricerca full-text. È possibile creare indici di testo a campo singolo o indici di testo composto con più di un campo di testo. Tuttavia, è possibile avere un solo indice di testo per raccolta. Per ulteriori informazioni, consulta Esecuzione di ricerche di testo con Amazon DocumentDB.

Proprietà dell'indice supportate

Opzione	3.6	4.0	5.0	8.0	Cluster elastico
name	No	No	Sì	Sì	No

Creazione di un indice di testo

Utilizzate il `createIndex()` metodo per creare un indice di testo. La sintassi del metodo è: `db.collection.createIndex(<keys>, <options>)`

Il `keys` parametro è un documento JSON che specifica i campi e il tipo di indice di testo:

```
{
  "<field 1>": "text",
  "<field 2>": "text"
}
```

Il `options` parametro è un documento JSON che specifica il nome dell'indice:

```
{
  "name": "<name>"
}
```

Vedi Proprietà dell'indice per esempi di creazione di indici di testo.

Indici geospaziali

Gli indici geospaziali sono un tipo specializzato di indice progettato per interrogare e gestire in modo efficiente i dati geospaziali archiviati in una raccolta di documenti. Amazon DocumentDB supporta gli indici 2dsphere, progettati specificamente per gestire dati geospaziali su una sfera (come la Terra). Ciò consente calcoli e interrogazioni accurati basati sulla geometria sferica.

Gli indici geospaziali sono utili quando le applicazioni devono eseguire query basate sulla posizione, come ad esempio:

- trovare punti di interesse nelle vicinanze,
- determinare se un luogo rientra in un'area specifica
- calcolo delle distanze tra le sedi

Proprietà dell'indice supportate

Opzione	3.6	4.0	5.0	8.0	Cluster elastico
name	Sì	Sì	Sì	Sì	Sì

Creazione di un indice geospaziale

Usa il `createIndex()` metodo per creare un indice geospaziale. La sintassi del metodo è: `db.collection.createIndex(<key>, <options>)`

Il `key` parametro è un documento JSON che specifica il campo e il tipo di indice 2dsphere:

```
{
  "<field>": "2dsphere"
}
```

Il `options` parametro è un documento JSON che specifica le opzioni per l'indice:

```
{
  "name": "<name>"
}
```

Vedi Proprietà dell'indice per esempi di creazione di indici geospaziali.

Indici vettoriali

Gli indici vettoriali sono un tipo specializzato di indice progettato per interrogare e gestire in modo efficiente i dati vettoriali archiviati in una raccolta di documenti. Amazon DocumentDB supporta gli indici Hierarchical Navigable Small World (HNSW) e Inverted File with Flat Compression (IVFFlat).

Per ulteriori informazioni, consulta Ricerca vettoriale per Amazon DocumentDB.

Gli indici vettoriali sono utili per i casi d'uso dell'apprendimento automatico e dell'intelligenza artificiale generativa, come:

- ricerca semantica
- raccomandazione sul prodotto

- personalizzazione
- chatbot
- rilevamento delle frodi
- rilevamento di anomalie

Proprietà dell'indice supportate

Opzione	3.6	4.0	5.0	8.0	Cluster elastico
name	No	No	Sì	Sì	No

Creazione di un indice vettoriale

Usa il comando createIndex con il `runCommand()` metodo per creare un indice vettoriale. La sintassi è:

```
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]
    }
  }]
})
```

Il parametro chiave è un documento JSON che specifica il campo e il tipo di indice vettoriale:

```
{
```

```
  "<field>": "vector"
}
```

Vedi Proprietà dell'indice per esempi di creazione di indici vettoriali.

Proprietà dell'indice

Le proprietà degli indici definiscono il comportamento e le caratteristiche degli indici in Amazon DocumentDB. La comprensione di queste proprietà ti aiuta a progettare indici efficienti che soddisfino i requisiti specifici della tua applicazione.

Argomenti

- Proprietà dell'indice: nome
- Proprietà dell'indice: unica
- Proprietà dell'indice: sparse
- Proprietà dell'indice: parziale FilterExpression
- Proprietà dell'indice: scadere AfterSeconds
- Proprietà dell'indice: nascosta

Proprietà dell'indice: nome

Tipi di indice supportati

Opzione	3.6	4.0	5.0	8.0	Cluster elastico
campo singolo	Sì	Sì	Sì	Sì	Sì
compound	Sì	Sì	Sì	Sì	Sì
chiave multipla	Sì	Sì	Sì	Sì	Sì
testo	No	No	Sì	Sì	No
geospaziale	Sì	Sì	Sì	Sì	Sì

Opzione	3.6	4.0	5.0	8.0	Cluster elastico
vettore	No	No	Sì	Sì	No

Utilizzate l'opzione name per fornire un nome opzionale per l'indice.

Tutti gli esempi utilizzano il seguente documento di esempio:

```
{
  "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')
}
```

Campo singolo

```
db.collection.createIndex(
```

```
  {
    "productId": 1
  },
  {
    "name": "single_field_index"
  }
)
```

Composte

```
db.collection.createIndex(
  {
    "productId": 1,
    "manufacturer": 1
  },
  {
    "name": "compound_index"
  }
)
```

Multi-key

```
db.collection.createIndex(
  {
    "tags": 1
  },
  {
    "name": "multikey_index"
  }
)
```

Testo

```
db.collection.createIndex(
  {
    "name": "text"
  },
  {
    "name": "text_index"
  }
)
```

Dati geospaziali

```
db.collection.createIndex(
  {
    "supplier.location": "2dsphere"
  },
  {
    "name": "geospatial_index"
  }
)
```

Vettore

```
db.runCommand({
  "createIndexes": "collection",
  "indexes": [{
    "key": {
      "productEmbedding": "vector"
    },
    "name": "hnsw_index",
    "vectorOptions": {
      "type": "hnsw",
      "dimensions": 2,
      "similarity": "euclidean",
      "m": 16,
      "efConstruction": 64
    }
  }]
})
```

Proprietà dell'indice: unica

Tipi di indice supportati

Opzione	3.6	4.0	5.0	8.0	Cluster elastico
campo singolo	Sì	Sì	Sì	Sì	Sì
compound	Sì	Sì	Sì	Sì	Sì

Opzione	3.6	4.0	5.0	8.0	Cluster elastico
chiave multipla	Sì	Sì	Sì	Sì	Sì

Utilizza l'opzione unica per garantire l'unicità dei campi tra i documenti di una raccolta.

Esempi

Gli esempi seguenti mostrano come creare indici univoci nel seguente documento di esempio:

```
{
  "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')
}
```

campo singolo

Crea un indice univoco a campo singolo su ProductID per garantire che lo stesso ProductID non esista in più di un documento:

```
db.collection.createIndex(
  {
    "productId": 1
  },
  {
    name: "productId_unique",
    unique: true
  }
)
```

Composto

Crea un indice composto univoco su sku e manufacturer per assicurarti che la stessa combinazione di sku e manufacturer non esista in più di un documento:

```
db.collection.createIndex(
  {
    "sku": 1,
    "manufacturer": 1
  },
  {
    name: "sku_and_manufacturer_unique",
    unique: true
  }
)
```

chiave multipla

Crea un indice univoco a più chiavi sui codici a barre per garantire che qualsiasi valore nella matrice di codici a barre non esista in più di un documento:

```
db.collection.createIndex(
  {
    "barcodes": 1
  },
  {
```

```
    name: "barcodes_unique",
    unique: true
  }
)
```

L'indicizzazione di un array crea una voce di indice per ogni elemento dell'array. Ad esempio, se un array contiene 50 elementi, ha 50 voci di indice. Per questo motivo, gli indici univoci a più chiavi impongono l'unicità di tutti i singoli elementi. Ad esempio, i seguenti documenti violeranno il vincolo univoco sull'indice del campo dell'array di valori:

```
{ "values": [ 1, 2, 3] }
{ "values": [ 3, 2 ] }   --> 3 and 2 already exist
{ "values": [ 1 ] }      --> 1 already exists
```

Nota il seguente comportamento con indici univoci:

1. Se crei un indice univoco su dati esistenti in cui due (o più) documenti hanno gli stessi valori per i campi indicizzati, la creazione dell'indice fallirà con il seguente errore: `could not create unique index: <collection> index: <index name>`
2. Se si inserisce un documento in cui il valore del campo indicizzato corrisponde al valore di quel campo in un altro documento, l'inserimento avrà esito negativo e verrà visualizzato il seguente errore: `E11000 duplicate key error collection: <collection> index: <index name>`
3. Se si aggiorna un documento esistente in modo che il nuovo valore del campo indicizzato corrisponda al valore di quel campo in un altro documento, l'aggiornamento avrà esito negativo e verrà visualizzato il seguente errore: `E11000 duplicate key error collection: <collection> index: <index name>`
4. Se il campo indicizzato non è presente in un documento, il valore verrà considerato nullo. La creazione, l'inserimento e l'aggiornamento dell'indice falliranno come descritto sopra se il campo indicizzato non è presente in due (o più) documenti.

Proprietà dell'indice: sparse

Tipi di indice supportati

Opzione	3.6	4.0	5.0	8.0	Cluster elastico
campo singolo	Sì	Sì	Sì	Sì	Sì
compound	Sì	Sì	Sì	Sì	Sì
chiave multipla	Sì	Sì	Sì	Sì	Sì

Usa l'opzione sparse per saltare l'indicizzazione dei documenti privi dei campi indicizzati, riducendo le dimensioni dell'indice e risparmiando spazio in memoria. Poiché la dimensione dell'indice è inferiore, le query che lo utilizzano sono più efficienti. Affinché una query utilizzi un indice sparso, è necessario utilizzare la clausola $exists nei campi indicizzati. Se ometti la clausola $exists, Amazon DocumentDB non utilizzerà l'indice sparse.

Esempi

I seguenti esempi mostrano come creare indici sparsi nel seguente documento di esempio:

```
{
  "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')
}
```

Nota che le recensioni, il materiale e i campi di colore non sono presenti in tutti i documenti.

campo singolo

Crea un indice a campo singolo sparso nel campo del materiale:

```
db.collection.createIndex(
  {
    "material": 1
  },
  {
    "name": "material_sparse",
    "sparse": true
  }
)
```

Questo indice verrà utilizzato per trovare tutti i prodotti che hanno un valore per il materiale:

```
db.collection.find({
  "material": {
    $exists: true
  }
})
```

composto

Crea un indice composto sparso nei campi del materiale e del colore:

```
db.collection.createIndex(
```

```
  {
    "material": 1,
    "color": 1
  },
  {
    "name": "material_and_color_sparse",
    "sparse": true
  }
)
```

Questo indice verrà utilizzato per trovare tutti i prodotti che hanno una combinazione di valori di materiale e colore:

```
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
      }
    }
  ]
})
```

chiave multipla

Crea un indice a più chiavi sparso nell'array delle recensioni:

```
db.collection.createIndex(
  {
    "reviews": 1
  },
  {
    "name": "reviews_sparse",
    "sparse": true
  }
)
```

Questo indice verrà utilizzato per trovare tutti i prodotti che hanno un array di recensioni:

```
db.collection.find({
  "reviews": {
    $exists: true
  }
})
```

Proprietà dell'indice: parziale FilterExpression

Tipi di indice supportati

Opzione	3.6	4.0	5.0	8.0	Cluster elastico
campo singolo	No	No	Sì	Sì	No
compound	No	No	Sì	Sì	No
chiave multipla	No	No	Sì	Sì	No
geospaziale	No	No	No	No	No

Utilizzate l'FilterExpression opzione partial per creare un indice parziale che includa solo i documenti che soddisfano una condizione di filtro specificata. Ciò consente di creare indici più efficienti indicizzando solo un sottoinsieme di documenti in una raccolta, anziché indicizzare tutti i documenti, riducendo le dimensioni dell'indice e risparmiando spazio in memoria. Poiché la dimensione dell'indice è inferiore, le query che lo utilizzano sono più efficienti. Amazon DocumentDB utilizzerà l'indice parziale nei seguenti scenari:

- Il predicato della query corrisponde esattamente all'espressione del filtro dell'indice parziale.
- Il risultato previsto del filtro di interrogazione è un sottoinsieme logico del filtro parziale.
- Un sottopredicato della query può essere utilizzato insieme ad altri indici.

Per ulteriori informazioni, consulta Indice parziale.

Esempi

Gli esempi seguenti mostrano come creare indici parziali nel seguente documento di esempio:

```
{
  "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')
}
```

campo singolo

Crea un indice parziale a campo singolo relativo al produttore per i prodotti con il colore Space Grey:

```
db.collection.createIndex(
  {
    "manufacturer": 1
  },
  {
    "name": "manufacturer_space_gray",
    "partialFilterExpression": {
      "color": {
        $eq: "Space Gray"
      }
    }
  }
)
```

Questo indice verrà utilizzato per trovare prodotti a colori Space Gray suddivisi per produttore:

```
db.collection.find({
  "manufacturer": "The Manufacturer",
  "color": {
    $eq: "Space Gray"
  }
})
```

composto

Crea un indice composto parziale sul produttore e sul colore per i prodotti che hanno un materiale in policarbonato:

```
db.collection.createIndex(
  {
    "manufacturer": 1,
    "color": 1
  },
  {
    "name": "manufacturer_and_color_polycarbonate",
    "partialFilterExpression": {
      "material": {
        $eq: "Polycarbonate"
      }
    }
  }
)
```

Questo indice verrà utilizzato per trovare prodotti in materiale policarbonato per produttore e colore:

```
db.collection.find({
  "manufacturer": "The Manufacturer",
  "color": "Space Gray",
  "material": {
    $eq: "Polycarbonate"
  }
})
```

chiave multipla

Crea un indice parziale a più chiavi sui tag dei prodotti il cui produttore è The Manufacturer:

```
db.collection.createIndex(
```

```
  {
    "tags": 1
  },
  {
    "name": "tags_the_manufacturer",
    "partialFilterExpression": {
      "manufacturer": {
        $eq: "The Manufacturer"
      }
    }
  }
)
```

Questo indice verrà utilizzato per trovare prodotti etichettati con una stampante il cui produttore è The Manufacturer:

```
db.collection.find({
  "tags": "printer",
  "manufacturer": {
    $eq: "The Manufacturer"
  }
})
```

Proprietà dell'indice: scadere AfterSeconds

Tipi di indice supportati

Tipo di indice	3.6	4.0	5.0	8.0	Cluster elastico
campo singolo	Sì	Sì	Sì	Sì	Sì
chiave multipla	Sì	Sì	Sì	Sì	Sì

Usa l'AfterSeconds opzione expire per creare un indice Time to Live (TTL). Gli indici TTL consentono di eliminare i documenti in base alla loro età in base alla condizione di timeout per ciascun documento. Quando un documento raggiunge la data TTL specificata, viene eliminato dalla raccolta.

Per le procedure consigliate con le eliminazioni TTL, vedere. Carichi di lavoro TTL e serie temporali

Esempi

Gli esempi seguenti mostrano come creare indici TTL nel seguente documento di esempio:

```
{
  "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')
```

```
}
```

campo singolo

Crea un indice TTL su LastUpdated per eliminare i documenti che non sono stati aggiornati da 90 giorni:

```
db.collection.createIndex(
  {
    "lastUpdated": 1
  },
  {
    "name": "lastUpdated_ttl",
    "expireAfterSeconds": 7776000
  }
)
```

chiave multipla

Crea un indice TTL su reviews.review_date per eliminare i documenti che non hanno ricevuto recensioni nell'ultimo anno:

```
db.collection.createIndex(
  {
    "reviews.review_date": 1
  },
  {
    "name": "reviews_review_date_ttl",
    "expireAfterSeconds": 31536000
  }
)
```

Nota che se l'indice TTL si trova su un campo di matrice, vengono controllati tutti gli elementi dell'array. Se un timestamp nell'array soddisfa la condizione di timeout, il documento verrà eliminato.

Proprietà dell'indice: nascosta

Amazon DocumentDB supporta la proprietà dell'`hidden`indice a partire da Amazon DocumentDB 8.0.1+ e Amazon DocumentDB 5.0.2+.

Gli indici nascosti non sono visibili al Query Planner e non possono essere utilizzati per supportare una query. Nascondendo un indice dal planner, puoi valutare il potenziale impatto dell'eliminazione

dell'indice senza eliminarlo effettivamente. Se l'impatto è negativo, puoi mostrare l'indice invece di dover ricreare un indice eliminato.

La `hidden` proprietà è supportata su tutti i tipi di indice, inclusi gli indici a campo singolo, composto, multichiave, di testo, geospaziale e vettoriale.

Comportamento

Oltre ad essere nascosti dal planner, gli indici nascosti sono completamente gestiti e applicati. Ad esempio:

- Se un indice nascosto è un indice univoco, l'indice applica comunque il suo vincolo univoco ai documenti.
- Se un indice nascosto è un indice TTL, l'indice scade comunque con i documenti.
- Gli indici nascosti sono inclusi nei risultati. `listIndexes` `db.collection.getIndexes()`
- Gli indici nascosti vengono aggiornati durante le operazioni di scrittura sulla raccolta e continuano a consumare spazio su disco e memoria. Di conseguenza, sono inclusi in operazioni statistiche come `db.collection.stats()` e`$indexStats`.
- Se si nasconde un indice non nascosto o si mostra un indice nascosto, si ripristina il relativo. `$indexStats` Nascondere un indice già nascosto o mostrare un indice già nascosto non viene reimpostato. `$indexStats`

Restrizioni

- Non è possibile nascondere l'indice. `_id`
- Non è possibile utilizzare `cursor.hint()` per forzare l'uso di un indice nascosto.
- `$merge`ha esito negativo se l'indice univoco richiesto nei `on` campi è nascosto. Rende visibile l'indice prima dell'esecuzione`$merge`.

Esempi

Crea un indice nascosto

Per creare un indice nello stato nascosto, utilizzate il `db.collection.createIndex()` metodo e impostate l'`hidden`opzione su `true` nella definizione dell'indice.

Ad esempio, la seguente operazione crea un indice ascendente nascosto sul `productId` campo:

```
db.collection.createIndex(
  { "productId": 1 },
  { "name": "productId_1", "hidden": true }
)
```

Per verificare, esegui `db.collection.getIndexes()` o `listIndexes` sulla raccolta. Un indice nascosto include `"hidden": true` nella sua definizione.

```
db.collection.getIndexes()
[
  {
    "v": 4,
    "key": { "_id": 1 },
    "name": "_id_"
  },
  {
    "v": 4,
    "key": { "productId": 1 },
    "name": "productId_1",
    "hidden": true
  }
]
```

L'opzione index `hidden` viene restituita solo se il valore è`true`.

Nascondi un indice esistente

Utilizzate il `collMod` comando per nascondere un indice esistente. Identifica l'indice in base al suo valore `name` o al suo `keyPattern` valore e impostalo `hidden` su`true`.

L'esempio seguente nasconde un indice in base al nome.

```
db.runCommand({
  collMod: "collection",
  index: {
    name: "productId_1", // Specify the index name
    hidden: true
  }
})
```

È inoltre possibile identificare l'indice in base al relativo schema chiave.

```
db.runCommand({
  collMod: "collection",
  index: {
    keyPattern: { productId: 1 }, // Specify the index key specification document
    hidden: true
  }
})
```

Mostra un indice esistente

Per rendere nuovamente visibile un indice nascosto, usa il `collMod` comando with `hidden` set to`false`.

```
db.runCommand({
  collMod: "collection",
  index: { name: "productId_1", hidden: false }
})
```

Poiché gli indici vengono mantenuti completamente quando sono nascosti, l'indice è immediatamente disponibile per l'uso una volta aperto.

Utilizzo dei flussi di modifiche con Amazon DocumentDB

La funzionalità dei flussi di modifica di Amazon DocumentDB (con compatibilità con MongoDB) fornisce una sequenza ordinata nel tempo di eventi di modifica che si verificano all'interno delle raccolte del cluster. Puoi leggere gli eventi da un flusso di modifica per implementare molti casi d'uso diversi, inclusi i seguenti:

- Notifica di modifica
- Full-text cerca con Amazon Service (Service) OpenSearch OpenSearch
- Analisi con Amazon Redshift

Le applicazioni possono utilizzare i flussi di modifica per sottoscrivere tutte le modifiche ai dati su singole raccolte. Gli eventi di flussi di modifica vengono ordinati man mano che si verificano nel cluster e vengono archiviati per 3 ore (valore predefinito) dopo la registrazione dell'evento. Il periodo di conservazione può essere esteso fino a 7 giorni utilizzando il `change_stream_log_retention_duration` parametro. Per modificare il periodo di

conservazione del flusso di modifiche, consultaModifica della durata di conservazione del registro del flusso di modifiche.

Argomenti

- Operazioni supportate
- Eventi estesi
- Fatturazione
- Limitazioni
- Abilitazione dei flussi di modifica
- Esempio: utilizzo dei flussi di modifica con Python
- Ricerca completa del documento
- Usare $change StreamSplitLargeEvent per dividere eventi di grandi dimensioni in frammenti
- Ripresa di un flusso di modifiche
- Ripresa di un flusso di modifiche con start AtOperationTime
- Ripresa di un flusso di modifiche con post BatchResumeToken
- Transazioni nei flussi di modifica
- Modifica della durata di conservazione del registro del flusso di modifiche
- Utilizzo dei flussi di modifica su istanze secondarie

Operazioni supportate

Amazon DocumentDB supporta le seguenti operazioni per i flussi di modifiche:

- Tutti gli eventi di modifica supportati in MongoDB `db.collection.watch()` e API. `db.watch()` `client.watch()`
- Ricerca completa dei documenti per gli aggiornamenti.
- Fasi di aggregazione:`$match`,,`$project`, `$redact``$addFields`, `$replaceRoot` e. `$changeStreamSplitLargeEvent`
- Ripresa di un flusso di modifiche da un token di curriculum
- Ripresa di un flusso di modifiche da un timestamp utilizzando `startAtOperation` (applicabile ad Amazon DocumentDB 4.0+)
- Notifica di eventi DDL per l'`createCollection`utilizzo del `showExpandedEvents` parametro (applicabile ad Amazon DocumentDB 8.0 a partire dalla versione del motore 8.0.2).

- Notifica di evento DDL per l'`createIndexes`utilizzo del `showExpandedEvents` parametro (applicabile ad Amazon DocumentDB 8.0 a partire dalla versione del motore 8.0.2).

Eventi estesi

> **Note**
>
> Gli eventi estesi sono disponibili in Amazon DocumentDB 8.0 a partire dalla versione del motore 8.0.2.

I flussi di modifica supportano le notifiche degli eventi DDL (Data Definition Language) e. `createCollection` `createIndexes` Per includere eventi estesi, apri un cursore di modifica del flusso di modifica con l'opzione. `showExpandedEvents`

Ad esempio:

mongosh Method

```
let cur = db.names.watch( [ ], {
   showExpandedEvents: true
} )
cur.next()
```

Aggregation Stage

```
let cur = db.names.aggregate( [
   { $changeStream: { showExpandedEvents: true } }
] )
cur.next()
```

Fatturazione

La funzionalità Change Streams di Amazon DocumentDB è disabilitata per impostazione predefinita e non comporta costi aggiuntivi finché non viene abilitata. L'utilizzo dei flussi di modifica in un cluster comporta costi iOS di lettura e scrittura aggiuntivi e di storage. Puoi utilizzare l'operazione `modifyChangeStreams` API per abilitare questa funzionalità per il tuo cluster. Per ulteriori informazioni sui prezzi, consulta i prezzi di Amazon DocumentDB.

Limitazioni

I flussi di modifiche presentano le seguenti limitazioni in Amazon DocumentDB:

- In Amazon DocumentDB 3.6. e Amazon DocumentDB 4.0, i flussi di modifica possono essere aperti solo da una connessione all'istanza primaria di un cluster Amazon DocumentDB. La lettura dai flussi di modifiche su un'istanza di replica non è supportata su Amazon DocumentDB 3.6. e Amazon DocumentDB 4.0. Quando si richiama l'operazione API `watch()`, è necessario specificare una preferenza di lettura `primary` per assicurare che tutte le letture siano indirizzate all'istanza primaria (consulta la sezione Esempio).
- In Amazon DocumentDB 5.0, i flussi di modifica possono essere aperti sia da istanze primarie che da istanze secondarie, inclusi i cluster globali. Puoi specificare una preferenza di lettura secondaria per reindirizzare i flussi di modifiche alle istanze secondarie. Consulta Utilizzo dei flussi di modifica su istanze secondarie la sezione per ulteriori best practice e limitazioni.
- Gli eventi scritti in un flusso di modifiche per una raccolta sono disponibili per un massimo di 7 giorni (l'impostazione predefinita è 3 ore). I dati dei flussi di modifiche vengono eliminati dopo la finestra della durata di conservazione del log, anche se non si sono verificate nuove modifiche.
- Un'operazione di scrittura di lunga durata su una raccolta come `updateMany` o `deleteMany` può temporaneamente bloccare la scrittura degli eventi dei flussi di modifica fino al completamento della lunga operazione di scrittura.
- Amazon DocumentDB non supporta il log delle operazioni di MongoDB (). `oplog`
- Con Amazon DocumentDB, devi abilitare esplicitamente i flussi di modifica su una determinata raccolta.
- Se la dimensione totale di un evento di change stream (inclusi i dati di modifica e il documento completo, se richiesto) è maggiore di`16 MB`, il client riscontrerà un errore di lettura nel flusso di modifiche. Per evitare ciò, utilizzate lo `$changeStreamSplitLargeEvent` stage per suddividere l'evento in frammenti.
- Il driver Ruby non è attualmente supportato quando si utilizza `db.watch()` e `client.watch()` con Amazon DocumentDB 3.6.
- L'output del `updateDescription` comando in change streams è diverso in Amazon DocumentDB rispetto a MongoDB quando il valore aggiornato del campo è uguale a quello precedente:
  - Amazon DocumentDB non restituisce un campo nell'`updateDescription`output se il campo fornito è specificato nel `$set` comando e il suo valore di destinazione è già uguale al valore di origine.

- MongoDB restituisce il campo nell'output, anche se il valore specificato è uguale al valore corrente.

Abilitazione dei flussi di modifica

Puoi abilitare i flussi di modifica di Amazon DocumentDB per tutte le raccolte all'interno di un determinato database o solo per raccolte selezionate. Di seguito sono riportati esempi di come abilitare i flussi di modifica per diversi casi d'uso utilizzando la shell mongo. Le stringhe vuote vengono trattate come caratteri jolly quando si specificano i nomi del database e della raccolta.

```
//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});
```

I flussi di modifica saranno abilitati per una raccolta se una delle seguenti condizioni è vera:

- Sia il database sia la raccolta sono abilitati in modo esplicito.
- Il database contenente la raccolta è abilitato.

- Tutti i database sono abilitati.

L'eliminazione di una raccolta da un database non disabilita i flussi di modifica per tale raccolta se il database padre dispone anche di flussi di modifiche abilitati o se tutti i database nel cluster sono abilitati. Se viene creata una nuova raccolta con lo stesso nome della raccolta eliminata, i flussi di modifica verranno abilitati per tale raccolta.

È possibile elencare tutti i flussi di modifiche abilitati del cluster utilizzando la fase della pipeline di aggregazione `$listChangeStreams`. Tutte le fasi di aggregazione supportate da Amazon DocumentDB possono essere utilizzate nella pipeline per ulteriori elaborazioni. Se una raccolta precedentemente abilitata è stata disabilitata, non verrà visualizzata nell'output `$listChangeStreams`.

```
//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:{}}));
```

Esempio: utilizzo dei flussi di modifica con Python

Di seguito è riportato un esempio di utilizzo di un flusso di modifiche di Amazon DocumentDB con Python a livello di raccolta.

```
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': '015daf99d40000000101000000100009025'},
'clusterTime': Timestamp(1571789268, 1),
'documentKey': {'_id': ObjectId('5daf9502ea258751778163d7')},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'update',
'updateDescription': {'removedFields': [], 'updatedFields': {'x': 2}}}
"""
```

Di seguito è riportato un esempio di utilizzo di un flusso di modifiche di Amazon DocumentDB con Python a livello di database.

```
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'}
"""
```

Ricerca completa del documento

L'evento di modifica dell'aggiornamento non include il documento completo, ma include solo la modifica apportata. Se il caso d'uso richiede il documento completo interessato da un aggiornamento, è possibile abilitare la ricerca completa del documento all'apertura del flusso.

Il documento `fullDocument` per un evento di aggiornamento del flusso di modifiche rappresenta la versione più recente del documento aggiornato al momento della ricerca del documento. Se si

sono verificate modifiche tra l'operazione di aggiornamento e la ricerca `fullDocument`, il documento `fullDocument` potrebbe non rappresentare lo stato del documento al momento dell'aggiornamento.

Per creare un oggetto stream con la ricerca degli aggiornamenti abilitata, usa questo esempio:

```
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()
```

L'output dell'oggetto stream sarà simile al seguente:

```
{'_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}}}
```

Usare $change StreamSplitLargeEvent per dividere eventi di grandi dimensioni in frammenti

> **Note**
>
> La StreamSplitLargeEvent fase $change è disponibile in Amazon DocumentDB 8.0 a partire dalla versione del motore 8.0.2.

Quando si utilizza lo `$changeStreamSplitLargeEvent` stage, gli eventi di modifica superiori a 16 MB vengono suddivisi in frammenti che vengono restituiti in sequenza tramite il cursore del flusso di modifiche. Gli eventi inferiori a 16 MB vengono restituiti non suddivisi. Lo stage divide un evento lungo i limiti dei campi di primo livello; non viene suddiviso in un singolo campo né viene suddiviso in documenti secondari. Per questo motivo, se un singolo campo di primo livello è a sua volta più grande di 16 MB, lo stage non può produrre un frammento valido e restituisce invece un errore.

Come best practice, consigliamo di evitarne l'uso, `$changeStreamSplitLargeEvent` quando possibile, riducendo le dimensioni dell'evento.

La `$changeStreamSplitLargeEvent` fase può essere utilizzata solo in una `$changeStream` pipeline e deve essere utilizzata una sola volta come ultima fase.

Ogni frammento ha il suo `resumeToken` e il suo. `postBatchResumeToken` La ripresa da un token di frammento non definitivo (utilizzando un normale token di curriculum o un postBatchResumeToken) fornisce il frammento successivo. Nel caso in cui il token corrisponda al frammento finale di un evento, il cursore del change stream restituisce l'evento successivo.

Ripresa di un flusso di modifiche

È possibile riprendere un flusso di modifiche successivamente utilizzando un token di ripresa, che è uguale al campo `_id` dell'ultimo documento dell'evento di modifica recuperato.

```
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}}
"""
```

Ripresa di un flusso di modifiche con start **`AtOperationTime`**

È possibile riprendere un flusso di modifiche in un secondo momento da un determinato timestamp utilizzando. `startAtOperationTime`

> **Note**
>
> La capacità di utilizzo `startAtOperationTime` è disponibile in Amazon DocumentDB 4.0+. Quando si utilizza`startAtOperationTime`, il cursore del flusso di modifiche restituirà solo le modifiche avvenute durante o dopo il timestamp specificato. I `resumeAfter` comandi `startAtOperationTime` and si escludono a vicenda e quindi non possono essere usati insieme.

```
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': '015f7e9502000000050100000005000fe038'},
'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}}
"""
```

Ripresa di un flusso di modifiche con post **`BatchResumeToken`**

Il flusso di modifiche di Amazon DocumentDB ora restituisce un campo aggiuntivo chiamato. `postBatchResumeToken` Questo campo viene restituito dal `$changestream` comando e dal `getMore` comando.

Esempio del `$changestream` comando in Python:

```
db.command({"aggregate": "sales", "pipeline": [{ "$changeStream": {}}], "cursor":
 {"batchSize": 1}})
```

Output previsto:

```
cursor" : {
   "firstBatch" : [ ],
   "postBatchResumeToken" : {"_data" : "0167c8cbe60000000004"},
   "id" : NumberLong("9660788144470"),
   "ns" : "test.sales"
}
```

Esempio del `getMore` comando in Python:

```
db.command({"getMore": NumberLong(<cursor id>), "collection": "sales", "batchSize":
 1 })
```

Output previsto

```
cursor" : {
   "nextBatch" : [ ],
   "postBatchResumeToken" : {"_data" : "0167c8cbe60000000004"},
   "id" : NumberLong("9660788144470"),
   "ns" : "test.sales"
}
```

Il `postBatchResumeToken` campo può essere utilizzato per aprire nuovi cursori del flusso di modifiche nel `resumeAfter` campo, in modo simile a come viene utilizzato il token di curriculum.

Apri uno stream che inizia dopo quello selezionato: `postBatchResumeToken`

```
post_batch_resume_token = output['cursor']['postBatchResumeToken']
stream = db.watch(full_document='updateLookup', resume_after=post_batch_resume_token)
```

A differenza di un normale token di curriculum che corrisponde sempre a una voce del registro delle operazioni (oplog) che riflette un evento effettivo, `postBatchResumeToken` corrisponde a una voce di oplog analizzata dal flusso di modifiche sul server, che non è necessariamente una modifica corrispondente.

Il tentativo di riprendere con un vecchio token di curriculum regolare costringerà il database a scansionare tutte le voci dell'olog comprese tra il timestamp specificato e l'ora corrente. Ciò può generare internamente molte interrogazioni, con la scansione di ogni sottoquery per un breve periodo

di tempo. Ciò causerà un picco nell'utilizzo della CPU e un peggioramento delle prestazioni del database. La ripresa con l'ultima opzione `postBatchResumeToken` salta la scansione delle voci di olog non corrispondenti.

Transazioni nei flussi di modifica

Gli eventi del flusso di modifiche non conterranno eventi derivanti da transazioni interrotte non salvate and/or . Ad esempio, se avvii una transazione con un'`INSERT`operazione e un'`UPDATE`operazione e se l'`INSERT`operazione ha esito positivo, ma l'`UPDATE`operazione ha esito negativo, la transazione verrà annullata. Poiché questa transazione è stata annullata, il flusso di modifiche non conterrà alcun evento per questa transazione.

Modifica della durata di conservazione del registro del flusso di modifiche

È possibile modificare la durata di conservazione del registro del flusso di modifiche in modo che sia compresa tra 1 ora e 7 giorni utilizzando Console di gestione AWS o. AWS CLI

Using the Console di gestione AWS

Per modificare la durata di conservazione del registro del flusso di modifica

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nel riquadro Parameter groups (Gruppi di parametri) scegliere il gruppo di parametri cluster associato al cluster. Per identificare il gruppo di parametri del cluster associato al cluster, consulta Determinazione del gruppo di parametri di un cluster Amazon DocumentDB.
4. La pagina risultante mostra i parametri e i relativi dettagli corrispondenti per questo gruppo di parametri del cluster. Selezionare il parametro `change_stream_log_retention_duration`.

5. In alto a destra della pagina, scegliere Edit (Modifica) per modificare il valore del parametro. Il `change_stream_log_retention_duration` parametro può essere modificato per essere compreso tra 1 ora e 7 giorni.
6. Apportare la modifica, quindi scegliere Modify cluster parameter (Modifica parametro cluster) per salvare le modifiche. Per annullare le modifiche, selezionare Cancel (Annulla).

Using the AWS CLI

Per modificare il parametro `change_stream_log_retention_duration` di un gruppo di parametri del cluster, utilizzare l'operazione `modify-db-cluster-parameter-group` con i parametri seguenti.

- **`--db-cluster-parameter-group-name`**: obbligatorio. Il nome del gruppo di parametri del cluster che stai modificando. Per identificare il gruppo di parametri del cluster associato al cluster, consulta Determinazione del gruppo di parametri di un cluster Amazon DocumentDB.
- **`--parameters`**: obbligatorio. Il parametro che stai modificando. Ogni voce del parametro deve includere:
  - **`ParameterName`**— Il nome del parametro che state modificando. In questo caso, è `change_stream_log_retention_duration`
  - **`ParameterValue`**— Il nuovo valore per questo parametro.
  - **`ApplyMethod`**— Come si desidera che vengano applicate le modifiche a questo parametro. I valori consentiti sono `immediate` e `pending-reboot`.

> **Note**
>
> I parametri con `ApplyType` per `static` devono avere `ApplyMethod` per `pending-reboot`.

1. Per modificare i valori del parametro `change_stream_log_retention_duration`, eseguire il comando seguente e sostituire `parameter-value` con il valore a cui si desidera modificare il parametro.

   Per Linux, macOS o 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"
```

Per 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"
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "DBClusterParameterGroupName": "sample-parameter-group"
}
```

2. Attendi almeno 5 minuti.
3. Elenca i valori dei parametri di `sample-parameter-group` per essere certo che le modifiche siano state apportate.

   Per Linux, macOS o Unix:

   ```
   aws docdb describe-db-cluster-parameters \
       --db-cluster-parameter-group-name sample-parameter-group
   ```

   Per Windows:

   ```
   aws docdb describe-db-cluster-parameters ^
       --db-cluster-parameter-group-name sample-parameter-group
   ```

   L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

   ```
   {
       "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**
>
> La conservazione dei log di Change Stream non eliminerà i log più vecchi del `change_stream_log_retention_duration` valore configurato finché la dimensione del registro non sarà superiore a (>) 51.200 MB.

Utilizzo dei flussi di modifica su istanze secondarie

Per iniziare a utilizzare il change stream sulle istanze secondarie, apri il cursore del change stream con `readPreference` come secondario.

Puoi aprire un cursore del flusso di modifiche per controllare gli eventi di modifica su una raccolta specifica o su tutte le raccolte in un cluster o database. Puoi aprire un cursore del flusso di modifiche su qualsiasi istanza di Amazon DocumentDB e recuperare i documenti del flusso di modifiche dalle istanze di writer e reader. Puoi condividere i token del flusso di modifica (come `resumeToken` `ostartOperationTime`) tra diversi cursori del flusso di modifica aperti su un'istanza di writer e reader.

Esempio

```
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)
```

Linee guida e limitazioni per i flussi di modifiche su istanze secondarie

- Gli eventi del flusso di modifica devono essere replicati dall'istanza primaria alle istanze secondarie. Puoi monitorare il ritardo dalla `DBInstanceReplicaLag` metrica in Amazon. CloudWatch
- I timestamp sulle istanze secondarie potrebbero non essere sempre sincronizzati con l'istanza principale. In questo caso, aspettatevi ritardi nel timestamp dell'istanza secondaria in modo che possa recuperare il ritardo. Come procedura consigliata, utilizza `startAtOperationTime` o `resumeToken` per avviare l'orologio sull'istanza secondaria.
- Potresti riscontrare un throughput inferiore sulle istanze secondarie rispetto all'istanza principale se le dimensioni del documento sono elevate e il carico di lavoro di scrittura simultanea sull'istanza principale è elevato. `fullDocumentLookup` Come procedura consigliata, monitorate l'hit ratio della cache del buffer secondario e assicuratevi che il buffer cache hit ratio sia elevato.

Transazioni in Amazon DocumentDB

Amazon DocumentDB (con compatibilità MongoDB) supporta le transazioni in Amazon DocumentDB 4.0 e versioni successive. Puoi eseguire transazioni su più documenti, rendiconti, raccolte e database. Le transazioni semplificano lo sviluppo di applicazioni consentendo di eseguire operazioni atomiche, coerenti, isolate e durevoli (ACID) su uno o più documenti all'interno di un cluster Amazon DocumentDB. I casi d'uso più comuni per le transazioni includono l'elaborazione finanziaria, l'evasione e la gestione degli ordini e la creazione di giochi multigiocatore.

Non ci sono costi aggiuntivi per le transazioni. Paghi solo l'iOS in lettura e scrittura che utilizzi come parte delle transazioni.

Argomenti

- Requisiti
- Best practice
- Limitazioni
- Monitoraggio e diagnostica
- Livello di isolamento delle transazioni
- Casi d’uso
- Comandi supportati
- Funzionalità non supportate
- Sessioni
- Errori di transazione

Requisiti

Per utilizzare la funzionalità delle transazioni, devi soddisfare i seguenti requisiti:

- È necessario utilizzare Amazon DocumentDB 4.0 o versioni successive.
- È necessario utilizzare un driver compatibile con MongoDB 4.0 o successivo.

Best practice

Ecco alcune best practice per ottenere il massimo dalle transazioni con Amazon DocumentDB.

- Effettua o annulla sempre la transazione dopo che è stata completata. Lasciare una transazione in uno stato incompleto limita le risorse del database e può causare conflitti di scrittura.
- Si consiglia di limitare le transazioni al minor numero di comandi necessari. Se si dispone di transazioni con più dichiarazioni che possono essere suddivise in più transazioni più piccole, è consigliabile farlo per ridurre la probabilità di un timeout. Cerca sempre di creare transazioni brevi, non letture di lunga durata.

Limitazioni

- Amazon DocumentDB non supporta i cursori all'interno di una transazione.
- Amazon DocumentDB non può creare nuove raccolte in una transazione e non può query/update utilizzare raccolte inesistenti.
- Document-level i blocchi di scrittura sono soggetti a un timeout di 1 minuto, che non è configurabile dall'utente.
- Nelle versioni del motore precedenti alla 8.0.2, le scritture riutilizzabili non sono supportate. A partire dalla versione 8.0.2 del motore, le scritture riutilizzabili sono supportate al di fuori delle transazioni per le operazioni di scrittura in cui ogni singola scrittura ha effetto al massimo su un documento e le operazioni di commit e interruzione delle transazioni sono rieseguibili. Per ulteriori informazioni, consulta Retryable scrive in Amazon DocumentDB.
- Ogni istanza di Amazon DocumentDB ha un limite massimo al numero di transazioni simultanee che possono essere aperte sull'istanza. Per ulteriori informazioni, consulta Quote delle istanze.
- Per una determinata transazione, la dimensione del registro delle transazioni deve essere inferiore a 32 MB.
- Amazon DocumentDB supporta `count()` una transazione, ma non tutti i driver supportano questa funzionalità. Un'alternativa è utilizzare l'`countDocuments()`API, che traduce la query di conteggio in una query di aggregazione sul lato client.
- Le transazioni hanno un limite di esecuzione di un minuto e le sessioni hanno un timeout di 30 minuti. Se una transazione scade, verrà interrotta e qualsiasi comando successivo emesso all'interno della sessione per la transazione esistente genererà il seguente errore:

```
WriteCommandError({
"ok" : 0,
"operationTime" : Timestamp(1603491424, 627726),
"code" : 251,
"errmsg" : "Given transaction number 0 does not match any in-progress transactions."
```

```
}
```

Monitoraggio e diagnostica

Con il supporto per le transazioni in Amazon DocumentDB 4.0, sono state aggiunte CloudWatch metriche aggiuntive per aiutarti a monitorare le transazioni.

Nuove metriche CloudWatch

- `DatabaseTransactions`: il numero di transazioni aperte effettuate in un minuto.
- `DatabaseTransactionsAborted`: il numero di transazioni interrotte eseguite in un minuto.
- `DatabaseTransactionsMax`: il numero massimo di transazioni aperte in un periodo di un minuto.
- `TransactionsAborted`: il numero di transazioni interrotte su un'istanza in un periodo di un minuto.
- `TransactionsCommitted`: il numero di transazioni effettuate su un'istanza in un periodo di un minuto.
- `TransactionsOpen`: il numero di transazioni aperte su un'istanza eseguita in un minuto.
- `TransactionsOpenMax`: il numero massimo di transazioni aperte su un'istanza in un periodo di un minuto.
- `TransactionsStarted`: il numero di transazioni avviate su un'istanza in un periodo di un minuto.

> **Note**
>
> Per ulteriori CloudWatch parametri per Amazon DocumentDB, vai a. Monitoraggio di Amazon DocumentDB con CloudWatch

Inoltre, sono stati aggiunti nuovi campi a entrambi `currentOp` `lsid` e un nuovo stato per `«idle transaction»` e `serverStatus` transazioni:`currentActive`,`currentInactive`, `currentOpen` `totalAbortedtotalCommitted`, e. `transactionThreadId` `totalStarted`

Livello di isolamento delle transazioni

Quando si avvia una transazione, è possibile specificare `readConcern` sia il che `writeConcern` come mostrato nell'esempio seguente:

```
mySession.startTransaction({readConcern: {level: 'snapshot'}, writeConcern:
{w: 'majority'}});
```

Per impostazione `readConcern` predefinita, Amazon DocumentDB supporta l'isolamento delle istantanee. Se viene specificato un `readConcern` valore locale, disponibile o maggioritario, Amazon DocumentDB aggiornerà il `readConcern` livello a snapshot. Amazon DocumentDB non supporta il linearizzabile `readConcern` e la specificazione di tale problema di lettura genererà un errore.

Per impostazione predefinita`writeConcern`, Amazon DocumentDB supporta la maggioranza e il quorum di scrittura viene raggiunto quando quattro copie dei dati vengono mantenute in modo persistente su tre AZ. Se `writeConcern` viene specificato un valore inferiore, Amazon DocumentDB aggiornerà il valore alla maggioranza. `writeConcern` Inoltre, tutte le scritture di Amazon DocumentDB sono inserite nel journaling e il journaling non può essere disabilitato.

Casi d’uso

In questa sezione, esamineremo due casi d'uso per le transazioni: dichiarazione multipla e raccolta multipla.

Multi-Statement Transazioni

Le transazioni di Amazon DocumentDB sono costituite da più dichiarazioni, il che significa che puoi scrivere una transazione che comprende più istruzioni con un commit o un rollback esplicito. Puoi raggruppare`insert`, `update`e `delete`, e `findAndModify` azioni come un'unica operazione atomica.

Un caso d'uso comune per le transazioni con più dichiarazioni è una transazione di debito/credito. Ad esempio: devi dei soldi a un amico per dei vestiti. Pertanto, devi addebitare (prelevare) 500$ dal tuo conto e accreditare 500$ (deposito) sul conto del tuo amico. Per eseguire tale operazione, esegui sia le operazioni di debito che di credito all'interno di un'unica transazione per garantire l'atomicità. In questo modo si evitano scenari in cui 500$ vengono addebitati sul tuo conto, ma non accreditati sul conto del tuo amico. Ecco come si presenterebbe questo caso d'uso:

```
// *** 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 transazioni

Le nostre transazioni sono anche a raccolta multipla, il che significa che possono essere utilizzate per eseguire più operazioni all'interno di una singola transazione e su più incassi. Ciò fornisce una visione coerente dei dati e ne mantiene l'integrità. Quando esegui i comandi singolarmente<>, le transazioni sono esecuzioni «tutto o niente»: in questo caso, avranno successo o falliranno tutte.

Ecco un esempio di transazioni con più raccolte, che utilizzano lo stesso scenario e i dati dell'esempio per le transazioni con più dichiarazioni.

```
// *** 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
```

Esempi di API di transazione per l'API di callback

L'API di callback è disponibile solo per i driver 4.2+.

Javascript

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con Javascript.

```
// *** 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

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con. 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#

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con C#.

```
// 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

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con Ruby.

```
// 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

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con Go.

```
// 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

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con Java.

```
// 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

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con C.

```
// 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

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con Python.

```
// 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
```

Esempi di API di transazione per l'API principale

Javascript

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con Javascript.

```
// *** 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#

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con C#.

```
// 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

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con Ruby.

```
# 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

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con Go.

```
// 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

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con Java.

```
// 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

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con C.

```
// 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

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con Scala.

```
// 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

Il codice seguente dimostra come utilizzare l'API di transazione Amazon DocumentDB con Python.

```
// 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
```

Comandi supportati

Comando	Supportata
`abortTransaction`	Sì
`commitTransaction`	Sì
`endSessions`	Sì
`killSession`	Sì

Comando	Supportata
`killAllSession`	Sì
`killAllSessionsByPattern`	No
`refreshSessions`	No
`startSession`	Sì

Funzionalità non supportate

Metodi	Fasi o comandi

`db.collection.aggregate()`	`$collStats`
	`$currentOp`
	`$indexStats`
	`$listSessions`
	`$out`

`db.collection.count()`
`db.collection.countDocuments()`
	`$where`
	`$near`
	`$nearSphere`

`db.collection.insert()`	`insert`non è supportato se non viene eseguito su una raccolta esistente. Questo metodo è supportato se si rivolge a una raccolta preesiste nte.

Sessioni

Le sessioni MongoDB sono un framework utilizzato per supportare scritture ripetibili, coerenza causale, transazioni e gestire le operazioni tra database. Quando viene creata una sessione, il client genera un identificatore logico di sessione (lsid) che viene utilizzato per etichettare tutte le operazioni all'interno di quella sessione durante l'invio di comandi al server.

Amazon DocumentDB supporta l'uso di sessioni per abilitare le transazioni, ma non supporta la coerenza causale. A partire dalla versione 8.0.2 del motore, Amazon DocumentDB supporta scritture riutilizzabili al di fuori delle transazioni per le operazioni di scrittura in cui ogni singola scrittura influisce al massimo su un documento. Nelle versioni precedenti del motore, le scritture rieseguibili non sono supportate. Per ulteriori informazioni, consulta Retryable scrive in Amazon DocumentDB.

Quando si utilizzano transazioni all'interno di Amazon DocumentDB, una transazione verrà avviata dall'interno di una sessione utilizzando l'`session.startTransaction()`API e una sessione supporta una singola transazione alla volta. Allo stesso modo, le transazioni vengono completate utilizzando le API commit (`session.commitTransaction()`) o abort (). `session.abortTransaction()`

Coerenza causale

La coerenza causale garantisce che all'interno di una singola sessione client il client osservi la coerenza monatomica lettura dopo scrittura e che le scritture seguano le letture e queste garanzie si applicano a tutte le istanze di un cluster reads/writes, non solo a quella primaria. Amazon DocumentDB non supporta la coerenza causale e la seguente dichiarazione genererà un errore.

```
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()
```

Puoi disattivare la coerenza causale all'interno di una sessione. Tieni presente che così facendo potrai utilizzare il framework della sessione, ma non fornirai garanzie di coerenza causale per le letture. Quando si utilizza Amazon DocumentDB, le letture dal primario saranno coerenti lettura dopo scrittura e le letture dalle istanze di replica saranno infine coerenti. Le transazioni sono il caso d'uso principale per l'utilizzo delle sessioni.

```
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 }
```

Retryable scrive

Con Retryable Writes, il driver riprova automaticamente un'operazione di scrittura una volta se si verifica un errore di rete o se il driver non riesce a trovare il file primario. A partire dalla versione 8.0.2 del motore, Amazon DocumentDB supporta le scritture riutilizzabili. Nelle versioni precedenti del motore, le scritture riutilizzabili devono essere disabilitate impostando la stringa di connessione. `retryWrites=false` Per ulteriori informazioni, consulta Retryable scrive in Amazon DocumentDB.

Errori di transazione

Quando si utilizzano le transazioni, ci sono scenari che possono generare un errore che indica che il numero di transazione non corrisponde a nessuna transazione in corso.

L'errore può essere generato in almeno due diversi scenari:

- Dopo il timeout della transazione di un minuto.
- Dopo il riavvio di un'istanza (a causa di patch, crash recovery, ecc.), è possibile ricevere questo errore anche nei casi in cui la transazione è stata completata con successo. Durante il riavvio di un'istanza, il database non è in grado di distinguere tra una transazione completata con successo e una transazione interrotta. In altre parole, lo stato di completamento della transazione è ambiguo.

Il modo migliore per gestire questo errore è rendere idempotenti gli aggiornamenti transazionali, ad esempio utilizzando il `$set` mutatore anziché un'operazione. increment/decrement Vedi sotto:

```
{ "ok" : 0,
"operationTime" : Timestamp(1603938167, 1),
"code" : 251,
"errmsg" : "Given transaction number 1 does not match any in-progress transactions."
}
```

Retryable scrive in Amazon DocumentDB

A partire dalla versione 8.0.2 del motore, Amazon DocumentDB supporta le scritture rieseguibili. Quando un'operazione di scrittura fallisce a causa di un errore transitorio di rete o di una scelta primaria, il driver può riprovare automaticamente l'operazione esattamente una volta. Amazon DocumentDB deduplica il tentativo di scrittura in modo che l'operazione venga applicata al massimo una volta, preservando l'idempotenza.

Le scritture retryable richiedono un driver che supporti il protocollo Retryable Writes. MongoDB-compatible La maggior parte dei MongoDB-compatible driver attuali abilita le scritture riutilizzabili per impostazione predefinita (`retryWrites=true`nella stringa di connessione). Se state effettuando l'aggiornamento al motore 8.0.2 da una versione precedente, potete rimuoverlo `retryWrites=false` dalla stringa di connessione per abilitare questa funzionalità.

Requisiti

Per utilizzare le scritture riutilizzabili, è necessario soddisfare i seguenti requisiti:

- Motore Amazon DocumentDB versione 8.0.2 o successiva.
- Un MongoDB-compatible driver che supporta scritture riutilizzabili. Per la versione minima, consulta la documentazione del driver in uso.
- La stringa di connessione deve include`retryWrites=true`, oppure il driver deve utilizzare per impostazione predefinita le scritture ripetibili (la maggior parte dei driver attuali lo fa).

Operazioni supportate

Le scritture riutilizzabili si applicano alle operazioni di scrittura in cui ogni singola scrittura influisce al massimo su un documento. Le operazioni batch sono idonee perché ogni scrittura all'interno del batch viene deduplicata separatamente.

Le seguenti operazioni di scrittura possono essere ripetute:

- `insertOne`
- `insertMany`
- `updateOne`
- `deleteOne`
- `findOneAndUpdate`
- `findOneAndDelete`
- `findOneAndReplace`
- `bulkWrite`(se composto da`insertOne`, `updateOne`e`deleteOne`, o `replaceOne` operazioni)

Per `insertMany` e`bulkWrite`, ogni documento del batch viene deduplicato singolarmente. Se è necessario un nuovo tentativo, vengono inseriti solo i documenti non ancora applicati.

> **Note**
>
> `commitTransaction`e `abortTransaction` sono inoltre riutilizzabili, come comandi di controllo delle transazioni anziché come scritture. Le singole scritture all'interno di una transazione non sono riutilizzabili. Per ulteriori informazioni, consulta Limitazioni.

Limitazioni

Le seguenti limitazioni si applicano alle scritture rieseguibili in Amazon DocumentDB:

- `updateMany`e non sono `deleteMany` riutilizzabili.
- Le scritture all'interno di transazioni con più dichiarazioni non sono ripetibili. Le operazioni di conferma e interruzione delle transazioni possono essere ripetute separatamente.
- I documenti devono includere un `_id` campo per gli inserti riutilizzabili.
- Le scritture riutilizzabili sono disponibili solo nella versione del motore 8.0.2 e successive. Nelle versioni precedenti del motore, impostate la stringa `retryWrites=false` di connessione per evitare errori.
- Le operazioni di scrittura non possono essere riprovate sui cluster che utilizzano la versione 1.0 di Query Planner.

- Per e per `findOneAndUpdate` `findOneAndReplace` le operazioni che restituiscono documenti di grandi dimensioni, le scritture riutilizzabili possono aumentare la latenza di scrittura perché l'intero documento dei risultati viene memorizzato nella cache per la deduplicazione. `findOneAndDelete` Se le prestazioni di scrittura sono fondamentali e i documenti restituiti sono di grandi dimensioni, valuta la possibilità di impostare tali carichi di lavoro. `retryWrites=false`

Attivazione delle scritture riutilizzabili

Per abilitare le scritture riutilizzabili, includi `retryWrites=true` nella stringa di connessione:

```
mongodb://<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=true
```

Se il driver è impostato per impostazione predefinita`retryWrites=true`, puoi rimuovere qualsiasi elemento esplicito `retryWrites=false` dalla stringa di connessione.

Per la procedura dettagliata completa sulla stringa di connessione, consulta. Connessione programmatica ad Amazon DocumentDB

Come funzionano le scritture riutilizzabili

Quando l'applicazione invia una scrittura con`retryWrites=true`, il driver allega un ID di sessione logico (`lsid`) e un numero di transazione (`txnNumber`) al comando di scrittura. Amazon DocumentDB utilizza questi identificatori per deduplicare le scritture:

- Al primo tentativo, Amazon DocumentDB esegue la scrittura e memorizza nella cache il risultato.
- Se il driver riprova la stessa scrittura (stessa `lsid` `txnNumber`), Amazon DocumentDB restituisce il risultato memorizzato nella cache senza rieseguire la scrittura.
- Amazon DocumentDB rispetta i tentativi per 60 minuti dopo la scrittura originale. I driver riprovano immediatamente, quindi questa finestra è molto più lunga del necessario per un conducente.

Se un nuovo tentativo arriva più di 60 minuti dopo la scrittura originale, Amazon DocumentDB non ha più un risultato memorizzato nella cache ed esegue la scrittura come nuova operazione.

Gestione degli errori

I seguenti errori sono specifici delle scritture rieseguibili in Amazon DocumentDB:

Codice di errore	Nome	Description
225	TransactionTooOld	È già stato eseguito il commit di un numero di transazione più recente per la stessa sessione e la stessa dichiarazione, quindi questo nuovo tentativo è troppo vecchio per essere applicato.
301	Le scritture riutilizzabili non sono supportate	La scrittura includeva campi di scrittura riutilizzabili, ma la versione del motore non supporta le scritture rieseguibili o l'operazione non è rieseguibile. Impostato `retryWrites=false` quando ci si connette a versioni del motore precedenti alla 8.0.2.

Migrazione da RetryWrites=false

Se stai eseguendo l'aggiornamento alla versione 8.0.2 del motore e attualmente utilizzi nelle stringhe di connessione: `retryWrites=false`

1. Aggiorna il cluster alla versione del motore 8.0.2 o successiva.
2. `retryWrites=false`Rimuovi le stringhe di connessione dell'applicazione o modificale in. `retryWrites=true`
3. Non sono necessarie modifiche al codice dell'applicazione. Il conducente gestisce i tentativi automaticamente.

Utilizzo delle regole di confronto in Amazon DocumentDB 8.0

Amazon DocumentDB 8.0 ora supporta la collazione. La collazione consente di configurare regole specifiche della lingua per il confronto di stringhe. Con le regole di confronto, è possibile specificare regole per i confronti con distinzione tra maiuscole e minuscole o specificare una lingua locale.
Le regole di confronto possono essere configurate a livello di raccolta o di indice in DocumentDB 8.0. Quando si utilizza la collazione in DocumentDB, internamente viene creato un documento di confronto con i seguenti parametri.

```
{
locale: string,
caseLevel: boolean,
caseFirst: string,
strength: int,
numericOrdering: boolean,
alternate: string,
maxVariable: string,
backwards: boolean,
normalization: boolean
}
```

Limitazioni

Collation presenta le seguenti limitazioni in Amazon DocumentDB:

- Collation è compatibile con planner v3 disponibile in Amazon DocumentDB 8.0. Il passaggio a plannerv2 o plannerv1 può causare comportamenti incoerenti, inclusi gli errori «Index not found».
- A causa delle differenze intrinseche tra le librerie, se viene esportata una raccolta mongodb con regole di confronto, sarà necessario aggiornare i file metadata.bson e modificarne la versione dalla 57.1 alla 60.2 prima della migrazione.
- In rari casi, le impostazioni di confronto potrebbero violare il limite interno sul numero di caratteri, causando il seguente errore. «Errore: il documento di confronto ha più attributi non predefiniti di quelli supportati. Riduci il numero di opzioni.» In questo caso, prova a ridurre le opzioni fornite nel documento di confronto o, in alternativa, puoi provare a utilizzare i valori predefiniti laddove possibile.

Utilizzo delle viste in Amazon DocumentDB 8.0

- Amazon DocumentDB 8.0 ora supporta le visualizzazioni. Le viste funzionano come raccolte virtuali che presentano dati in base a operazioni di aggregazione specificate. Quando si crea una vista, si definisce una query che trasforma i dati da una o più raccolte di fonti. Amazon DocumentDB 8.0 esegue questa query ogni volta che si accede alla vista, senza consumare risorse di storage aggiuntive. A differenza delle raccolte standard, le viste in Amazon DocumentDB 8.0 non archiviano documenti su disco, il che le rende una soluzione efficiente per presentare dati trasformati o filtrati alle applicazioni. Per creare una vista in Amazon DocumentDB, puoi utilizzare il comando createView o il metodo helper db.createView ():

```
db.createView("viewName","sourceCollection",
[
    { $match: { status: "active" } },
    { $project: { _id: 1, name: 1, email: 1 } }
]
)
```

Questo crea una vista denominata «viewName» basata su «SourceCollection» che include solo documenti e progetti attivi, solo i campi _id, name ed email. Le viste in Amazon DocumentDB sono di sola lettura. Le operazioni di scrittura sulle viste restituiranno un errore. Per prestazioni ottimali con set di dati di grandi dimensioni, puoi strutturare le tue pipeline di visualizzazione per massimizzare l'efficienza. Per pipeline di aggregazione complesse, si consiglia di utilizzare la fase $match come prima fase o all'inizio della pipeline per ridurre il numero di documenti che le fasi successive devono elaborare, migliorando così le prestazioni delle query.

Best practice

Di seguito sono elencate alcune best practice da seguire con Views.

- Filtra in anticipo: utilizza $match Stages nelle prime fasi di visualizzazione delle pipeline per ridurre la quantità di dati elaborati.
- Evita le aggregazioni complesse: per viste ad accesso frequente con aggregazioni complesse, prendi in considerazione la creazione di una raccolta separata con risultati precalcolati che vengono aggiornati periodicamente.
- Pianificazione dell'indice: assicurati che i campi utilizzati nelle pipeline di visualizzazione, specialmente nelle operazioni $match e $sort, siano indicizzati correttamente nella raccolta di origine.
- Ottimizzazione delle query: utilizza il comando explain per capire come vengono eseguite le query di visualizzazione e ottimizzarle di conseguenza.
- Alternative per le visualizzazioni: date le differenze funzionali tra le viste di Amazon DocumentDB e MongoDB, prendi in considerazione l'utilizzo di raccolte regolari con aggiornamenti pianificati come alternativa alle viste in caso di limitazioni.

Compatibilità con gli operatori di aggregazione

Amazon DocumentDB supporta molti operatori di aggregazione nelle definizioni delle viste, pur continuando a espandere la compatibilità. Quando usi le viste, concentrati su questi operatori supportati:

- $match per filtrare i documenti
- $project per la selezione e la trasformazione dei campi
- $addFields per aggiungere campi calcolati
- $sort per ordinare i risultati
- $limit e $skip per l'impaginazione

Alcuni operatori specializzati come $currentOp, $replaceRoot e $geoEar attualmente lavorano su query di aggregazione diretta anziché visualizzare le definizioni.

Sfruttando indici e visualizzazioni

Le viste in Amazon DocumentDB 8.0 utilizzano gli indici della raccolta sottostante. Di conseguenza, non è possibile creare, eliminare o ricostruire gli indici direttamente su una vista. Tuttavia, indici ben progettati nella raccolta di sorgenti possono migliorare in modo significativo le prestazioni delle query di visualizzazione Di seguito sono riportati alcuni passaggi per ottimizzare le prestazioni delle query sulle viste:

- Assicurati che esistano indici appropriati nei campi di raccolta dei sorgenti utilizzati nella pipeline della vista, specialmente nelle operazioni $match e $sort
- Utilizzate il metodo explain () per analizzare i piani di esecuzione delle query e verificare l'utilizzo dell'indice. E.g., `db.viewName.find({...}).explain()`

Intelligenza artificiale generativa di Amazon DocumentDB

Amazon DocumentDB offre funzionalità per consentire ai modelli di machine learning (ML) e intelligenza artificiale generativa (AI) di funzionare con i dati archiviati in Amazon DocumentDB in tempo reale. I clienti non devono più perdere tempo a gestire un'infrastruttura separata, a scrivere codice per connettersi a un altro servizio e a duplicare i dati dal loro database principale.

Per ulteriori informazioni sull'intelligenza artificiale e su come AWS può supportare le tue esigenze di intelligenza artificiale, consulta questo What-is "» articolo.

Argomenti

- No-code apprendimento automatico con Amazon SageMaker AI Canvas
- Ricerca vettoriale per Amazon DocumentDB

No-code apprendimento automatico con Amazon SageMaker AI Canvas

Amazon SageMaker AI Canvas ti consente di creare AI/ML modelli personalizzati senza dover scrivere una sola riga di codice. Puoi creare modelli ML per casi d'uso comuni come regressione e previsione e accedere e valutare i modelli di base (FM) di Amazon Bedrock. Puoi anche accedere a FM pubblici da Amazon SageMaker AI JumpStart per la generazione di contenuti, l'estrazione e il riepilogo del testo per supportare soluzioni di intelligenza artificiale generativa.

Come creare modelli ML senza codice con AI Canvas SageMaker

Amazon DocumentDB ora si integra con Amazon SageMaker AI Canvas per abilitare l'apprendimento automatico (ML) senza codice con i dati archiviati in Amazon DocumentDB. Ora puoi creare modelli ML per esigenze di regressione e previsione e utilizzare modelli di base per il riepilogo e la generazione di contenuti utilizzando i dati archiviati in Amazon DocumentDB senza scrivere una sola riga di codice.

SageMaker AI Canvas fornisce un'interfaccia visiva che consente ai clienti di Amazon DocumentDB di generare previsioni senza richiedere alcuna AI/ML competenza o scrivere una sola riga di codice. I clienti possono ora avviare lo spazio di lavoro SageMaker AI Canvas da Console di gestione AWS, importare e unire i dati di Amazon DocumentDB per la preparazione dei dati e la formazione dei modelli. I dati in Amazon DocumentDB possono ora essere utilizzati in SageMaker AI Canvas per creare e potenziare modelli per prevedere l'abbandono dei clienti, rilevare frodi, prevedere guasti di manutenzione, prevedere metriche aziendali e generare contenuti. I clienti possono ora pubblicare e condividere ML-driven informazioni tra i team utilizzando l'integrazione nativa di SageMaker AI Canvas con Quick. Le pipeline di inserimento dei dati in SageMaker AI Canvas vengono eseguite su istanze secondarie di Amazon DocumentDB per impostazione predefinita, assicurando che le prestazioni delle applicazioni e dei carichi di lavoro di inserimento di SageMaker AI Canvas non siano ostacolate.

I clienti di Amazon DocumentDB possono iniziare a usare SageMaker AI Canvas accedendo alla nuova pagina della console Amazon DocumentDB No-Code ML e collegandosi a spazi di lavoro AI Canvas nuovi o disponibili. SageMaker

Configurazione del dominio AI e del profilo utente SageMaker

Puoi connetterti ai cluster Amazon DocumentDB da domini SageMaker AI in esecuzione solo in modalità VPC. Avviando un dominio SageMaker AI nel tuo VPC, puoi controllare il flusso di dati dai tuoi SageMaker ambienti AI Studio e Canvas. Ciò consente di limitare l'accesso a Internet, monitorare e ispezionare il traffico utilizzando funzionalità AWS di rete e sicurezza standard e connettersi ad altre AWS risorse tramite endpoint VPC. Fai riferimento a Amazon SageMaker AI Canvas Guida introduttiva e configurazione di Amazon SageMaker AI Canvas in un VPC senza accesso a Internet, disponibile nella Amazon SageMaker AI Developer Guide, per creare il tuo dominio SageMaker AI da connettere al tuo cluster Amazon DocumentDB.

Configurazione delle autorizzazioni di accesso IAM per Amazon DocumentDB e AI Canvas SageMaker

Un utente Amazon DocumentDB `AmazonDocDBConsoleFullAccess` collegato al ruolo e all'identità associati può accedere a. Console di gestione AWS Aggiungi le seguenti azioni al ruolo o all'identità sopra menzionati per fornire l'accesso all'apprendimento automatico senza codice con Amazon SageMaker AI Canvas.

```
"sagemaker:CreatePresignedDomainUrl",
"sagemaker:DescribeDomain",
"sagemaker:ListDomains",
"sagemaker:ListUserProfiles"
```

Creazione di utenti e ruoli del database per SageMaker AI Canvas

Puoi limitare l'accesso alle azioni che gli utenti possono eseguire sui database utilizzando il controllo degli accessi basato sui ruoli (RBAC) in Amazon DocumentDB. RBAC funziona concedendo uno o più ruoli a un utente. Questi ruoli determinano le operazioni che un utente può eseguire sulle risorse del database.

Come utente Canvas, ti connetti a un database Amazon DocumentDB con credenziali di nome utente e password. Puoi creare un database user/role per un utente Canvas che ha accesso in lettura ai database specifici utilizzando la funzionalità RBAC di Amazon DocumentDB.

Ad esempio, utilizza l'operazione: `createUser`

```
db.createUser({
```

```
user: "canvas_user",
pwd: "<insert-password>",
roles: [{role: "read", db: "sample-database-1"}]
})
```

Questo crea un file con `canvas_user` autorizzazioni di lettura per il `sample-database-1` database. I tuoi analisti Canvas possono utilizzare questa credenziale per accedere ai dati nel tuo cluster Amazon DocumentDB. Fai riferimento a per saperne di Accesso al database tramite Access Control Role-Based più.

Regioni disponibili

L'integrazione senza codice è disponibile nelle regioni in cui sono supportati sia Amazon DocumentDB che Amazon SageMaker AI Canvas. Le regioni includono:

- us-east-1 (Virginia settentrionale)
- us-east-2 (Ohio)
- us-west-2 (Oregon)
- ap-northeast-1 (Tokyo)
- ap-northeast-2 (Seoul)
- ap-south-1 (Mumbai)
- ap-southeast-1 (Singapore)
- ap-southeast-2 (Sydney)
- eu-central-1 (Francoforte)
- eu-west-1 (Irlanda)

Fai riferimento ad Amazon SageMaker AI Canvas nella Amazon SageMaker AI Developer Guide per le ultime disponibilità nelle regioni.

Ricerca vettoriale per Amazon DocumentDB

La ricerca vettoriale è un metodo utilizzato nell'apprendimento automatico per trovare punti dati simili a un determinato punto dati confrontando le relative rappresentazioni vettoriali utilizzando metriche di distanza o somiglianza. Più i due vettori sono vicini nello spazio vettoriale, più simili sono considerati gli elementi sottostanti. Questa tecnica aiuta a catturare il significato semantico dei dati.

Questo approccio è utile in varie applicazioni, come i sistemi di raccomandazione, l'elaborazione del linguaggio naturale e il riconoscimento delle immagini.

La ricerca vettoriale per Amazon DocumentDB combina la flessibilità e la ricca capacità di interrogazione di un database di JSON-based documenti con la potenza della ricerca vettoriale. Se desideri utilizzare i tuoi dati Amazon DocumentDB esistenti o una struttura flessibile dei dati dei documenti per creare casi d'uso di machine learning e intelligenza artificiale generativa, come esperienza di ricerca semantica, consigli sui prodotti, personalizzazione, chatbot, rilevamento di frodi e rilevamento di anomalie, la ricerca vettoriale per Amazon DocumentDB è la scelta ideale per te. La ricerca vettoriale è disponibile nei cluster basati su istanze Amazon DocumentDB 5.0.

Argomenti

- Inserimento di vettori
- Creazione di un indice vettoriale
- Ottenere una definizione dell'indice
- Query sui vettori
- Caratteristiche e limitazioni
- Best practice

Inserimento di vettori

Per inserire vettori nel tuo database Amazon DocumentDB, puoi utilizzare i metodi di inserimento esistenti:

Esempio

Nell'esempio seguente, viene creata una raccolta di cinque documenti all'interno di un database di test. Ogni documento include due campi: il nome del prodotto e il corrispondente incorporamento vettoriale.

```
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]}
]);
```

Creazione di un indice vettoriale

Amazon DocumentDB supporta i metodi di indicizzazione Hierarchical Navigable Small World (HNSW) e Inverted File with Flat Compression (IVFFlat). Un indice IVFFlat separa i vettori in elenchi e successivamente cerca un sottoinsieme selezionato di quegli elenchi più vicini al vettore di query. D'altra parte, un indice HNSW organizza i dati vettoriali in un grafico a più livelli. Sebbene HNSW abbia tempi di compilazione più lenti rispetto a IVFFlat, offre migliori prestazioni di interrogazione e richiamo. A differenza di IVFFlat, HNSW non prevede alcuna fase di addestramento e consente di generare l'indice senza alcun caricamento iniziale dei dati. Per la maggior parte dei casi d'uso, utilizzate il tipo di indice HNSW per la ricerca vettoriale.

Se non crei un indice vettoriale, Amazon DocumentDB esegue una ricerca esatta del vicino più prossimo, garantendo un richiamo perfetto. Tuttavia, negli scenari di produzione, la velocità è fondamentale. Utilizzate indici vettoriali, che potrebbero sostituire alcuni richiami con una maggiore velocità. È importante notare che l'aggiunta di un indice vettoriale può portare a risultati di ricerca diversi.

Modelli

Puoi utilizzare quanto segue `createIndex` o i `runCommand` modelli per creare un indice vettoriale su un campo vettoriale:

Using createIndex

In alcuni driver, come mongosh e Java, l'utilizzo dei `vectorOptions` parametri in `createIndex` potrebbe generare un errore. In questi casi, usa: `runCommand`

```
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 alcuni driver, come mongosh e Java, l'utilizzo dei `vectorOptions` parametri in `createIndex` potrebbe generare un errore. In questi casi, usa: `runCommand`

```
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"
      }]
  }
);
```

Parametro	Requisito	Tipo di dati	Description	Valore/i
`name`	facoltativo	stringa	Specifica il nome dell'indice.	Carattere alfanumerico
`type`	facoltativo		Specifica il tipo di indice.	Supportato: hnsw o ifflat Predefinito: HNSW (patch del motore 3.0.4574 in poi)
`dimensions`	obbligatorio	intero	Specifica il numero di	Massimo 2.000 dimensioni.

Parametro	Requisito	Tipo di dati	Description	Valore/i
			dimensioni nei dati vettoriali.	
`similarity`	obbligatorio	stringa	Specifica la metrica della distanza utilizzata per il calcolo della somiglianza.	• **`euclidean`** • **`cosine`** • **`dotProduct`**
`lists`	obbligatorio per IVFFlat	intero	Specifica il numero di cluster utilizzati dall'indice IVFFlat per raggruppare i dati vettoriali. L'impostazione consigliata è il numero di documents/1000 per un massimo di 1 milione di documenti e per oltre 1 milione di documenti `. sqrt(# of documents)`	Minimo: 1 Numero massimo: consulta la tabella degli elenchi per tipo di istanza riportata di seguito. Caratteristiche e limitazioni
`m`	facoltativo	intero	Specifica il numero massimo di connessioni per un indice HNSW	Impostazione predefinita: 16 Intervallo [2, 100]

Parametro	Requisito	Tipo di dati	Description	Valore/i
`efConstruction`	facoltativo	intero	Specifica la dimensione dell'elenco dinamico di candidati per la costruzione del grafico per l'indice HNSW. `efConstruction` deve essere maggiore o uguale a (2 * m)	Impostazione predefinita: 64 Intervallo [4, 1000]

È importante impostare il valore dei sottoparametri come `lists` per IVFFlat `m` e `efConstruction` per HNSW in modo appropriato poiché influirà sui tempi di compilazione e sulle prestazioni della accuracy/recall ricerca. Un valore di elenco più elevato aumenta la velocità dell'interrogazione in quanto riduce il numero di vettori in ogni elenco, con conseguente riduzione delle aree. Tuttavia, una regione di dimensioni inferiori può comportare un maggior numero di errori di richiamo, con conseguente minore precisione. Per HNSW, l'aumento del valore `m` e della precisione, ma `efConstruction` aumenta anche il tempo e le dimensioni di creazione dell'indice. Fare riferimento agli esempi riportati di seguito:

Esempi

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
    }
  }
)
```

Ottenere una definizione dell'indice

Puoi visualizzare i dettagli dei tuoi indici, inclusi gli indici vettoriali, usando il comando: `getIndexes`

Esempio

```
db.collection.getIndexes()
```

Output di esempio

```
[
 {
  "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"
 }
]
```

Query sui vettori

Amazon DocumentDB supporta due operatori di ricerca vettoriale per interrogare i vettori:

Operatore di ricerca vettoriale classico

Usa il seguente modello per interrogare un vettore:

```
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]
      }
    }
  }
]);
```

Parametro	Requisito	Tipo	Description	Valore/i
vectorSearch	obbligatorio	operatore	Utilizzato all'interno del comando $search per	

Parametro	Requisito	Tipo	Description	Valore/i
			interrogare i vettori.	
vector	obbligatorio	array	Indica il vettore di interroga zione che verrà utilizzato per trovare vettori simili.	
path	obbligatorio	stringa	Definisce il nome del campo vettoriale.	
k	obbligatorio	intero	Specifica il numero di risultati restituiti dalla ricerca.	
similarity	obbligatorio	stringa	Specifica la metrica della distanza utilizzat a per il calcolo della somiglian za.	• **euclidean** • **cosine** • **dotProduct**

Parametro	Requisito	Tipo	Description	Valore/i
`probes`	facoltativo	intero	Il numero di cluster da ispezionare con la ricerca vettoriale. Un valore più alto offre un migliore richiamo a scapito della velocità. Può essere impostato sul numero di elenchi per la ricerca esatta del vicino più vicino (a quel punto il planner non utilizzerà l'indice) . L'imposta zione consiglia ta per iniziare la messa a punto è. `sqrt(# of lists)`	Impostazione predefinita: 1

Parametro	Requisito	Tipo	Description	Valore/i
`efSearch`	facoltativo	intero	Specifica la dimensione dell'elenco dinamico di candidati utilizzato dall'indice HNSW durante la ricerca. Un valore più elevato di `efSearch` consente un migliore richiamo a discapito della velocità.	Impostazione predefinita: 40 Intervallo [1, 1000]

È importante regolare con precisione il valore di `efSearch` (HNSW) o `probes` (IVFFlat) per ottenere le prestazioni e la precisione desiderate. Vedi i seguenti esempi di operazioni:

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
      }
    }
  }
]);
```

Output di esempio

L'aspetto dell'output di questa operazione è simile al seguente.

```
{ "_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 ] }
```

operatore $vectorSearch (disponibile in Amazon DocumentDB 8.0 e versioni successive)

Usa il seguente modello per interrogare un vettore:

```
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],
  }
}])
```

Caratteristiche e limitazioni

Compatibilità delle versioni

- La ricerca vettoriale per Amazon DocumentDB è disponibile solo sui cluster basati su istanze Amazon DocumentDB 5.0+.

Vettori

- Amazon DocumentDB può indicizzare vettori fino a 2.000 dimensioni. Tuttavia, è possibile memorizzare fino a 16.000 dimensioni senza indice.

Indici

- Per la creazione dell'indice IVFFlat, l'impostazione consigliata per il parametro degli elenchi è il numero di documenti per un massimo di 1 milione e documents/1000 `sqrt(# of documents)` per oltre 1 milione di documenti. A causa di un limite di memoria di lavoro, Amazon DocumentDB supporta un determinato valore massimo del parametro degli elenchi in base al numero di dimensioni. A titolo di riferimento, la seguente tabella fornisce i valori massimi dei parametri delle liste per vettori di 500, 1000 e 2.000 dimensioni:

Tipo di istanza	Elenchi con 500 dimensioni	Elenchi con 1000 dimensioni	Elenchi con 2000 dimensioni
t3.med	372	257	150
r5.l	915	741	511
r5.xl	1.393	1.196	901
r 5.2 xl	5.460	5.230	4.788
r 5.4xl	7.842	7.599	7.138
r 5,8 xl	11.220	10.974	10.498
r 5,12 xl	13.774	13.526	13.044
r 5,16 xl	15.943	15.694	15,208

Tipo di istanza	Elenchi con 500 dimensioni	Elenchi con 1000 dimensioni	Elenchi con 2000 dimensioni
r 5,24 xl	19.585	19,335	18.845

- Nessun'altra opzione di indice come `compound` `sparse` o è supportata con `partial` gli indici vettoriali.
- La creazione parallela dell'indice non è supportata per l'indice HNSW in Amazon DocumentDB 5.0.

Query vettoriale

- Per le query di ricerca vettoriale, è importante ottimizzare i parametri, ad esempio `probes` `efSearch` per ottenere risultati ottimali. Maggiore è il valore del `efSearch` parametro `probes` o, maggiore è il richiamo e minore è la velocità. L'impostazione consigliata per iniziare la regolazione fine del parametro delle sonde è. `sqrt(# of lists)`

Best practice

Scopri le best practice per lavorare con la ricerca vettoriale in Amazon DocumentDB. Questa sezione viene continuamente aggiornata man mano che vengono identificate nuove best practice.

- La creazione dell'indice Inverted File with Flat Compression (IVFFlat) prevede il clustering e l'organizzazione dei punti dati in base alle somiglianze. Affinché un indice sia più efficace, carica alcuni dati prima di creare l'indice.
- Per le query di ricerca vettoriale, è importante ottimizzare i parametri, ad esempio `probes` `efSearch` per ottenere risultati ottimali. Maggiore è il valore del `efSearch` parametro `probes` or, maggiore è il richiamo e minore è la velocità. L'impostazione consigliata per iniziare la regolazione fine del `probes` parametro è`sqrt(lists)`.

Risorse

- La ricerca vettoriale per Amazon DocumentDB è ora disponibile a livello generale sul blog AWS
- Esempio di codice di ricerca semantica
- Esempi di codice di ricerca vettoriale di Amazon DocumentDB

Utilizzo AWS Lambda con flussi di modifica

Amazon DocumentDB è integrato con, AWS Lambda quindi puoi utilizzare le funzioni Lambda per elaborare i record in un flusso di modifiche. La mappatura dell'origine degli eventi Lambda è una risorsa che può essere utilizzata per richiamare funzioni Lambda per elaborare eventi Amazon DocumentDB che non richiamano direttamente Lambda. Con Amazon DocumentDB change stream come fonte di eventi, puoi creare applicazioni basate sugli eventi che rispondono alle modifiche nei tuoi dati. Ad esempio, puoi utilizzare le funzioni Lambda per elaborare nuovi documenti, tenere traccia degli aggiornamenti ai documenti esistenti o registrare i documenti eliminati.

Puoi configurare una mappatura dell'origine degli eventi per inviare i record dal flusso di modifiche di Amazon DocumentDB a una funzione Lambda. Gli eventi possono essere inviati uno alla volta o raggruppati per una maggiore efficienza e verranno elaborati in ordine. È possibile configurare il comportamento in batch della mappatura dell'origine degli eventi in base alla durata di una finestra temporale specifica (0-300 secondi) o al numero di record in batch (limite massimo di 10.000 record). Puoi creare più mappature delle sorgenti di eventi per elaborare gli stessi dati con più funzioni Lambda o per elaborare elementi distinti da più stream con un'unica funzione.

Se la funzione restituisce un errore, Lambda riprova il batch finché non viene elaborato correttamente. Nel caso in cui gli eventi nel flusso di modifiche siano scaduti, Lambda disabiliterà la mappatura della sorgente degli eventi. In questo caso, puoi creare una nuova mappatura della sorgente degli eventi e configurarla con una posizione di partenza a tua scelta. Le mappature delle origini eventi Lambda elaborano gli eventi almeno una volta a causa della natura distribuita dei relativi poller. Di conseguenza, in rari casi, la tua funzione Lambda potrebbe ricevere eventi duplicati. Segui le migliori pratiche per lavorare con AWS Lambda le funzioni e crea funzioni idempotenti per evitare problemi legati agli eventi duplicati. Per ulteriori informazioni, consulta Using AWS Lambda console with Amazon DocumentDB nella Developer Guide. AWS Lambda

Come best practice in materia di prestazioni, la funzione Lambda deve essere di breve durata. Per evitare di introdurre ritardi di elaborazione non necessari, inoltre, non dovrebbe eseguire una logica complessa. In particolare, per un flusso a velocità elevata, è meglio attivare flussi di lavoro Step Function di post-elaborazione asincrona rispetto a funzioni Lambda sincrone a lunga durata. Per ulteriori informazioni AWS Lambda, consulta la AWS Lambda Developer Guide.

Limitazioni

Le seguenti sono limitazioni da considerare quando si lavora con Amazon DocumentDB e AWS Lambda:

- AWS Lambda è attualmente supportato solo su Amazon DocumentDB 4.0 e 5.0.
- AWS Lambda non è attualmente supportato su cluster elastici o cluster globali.
- AWS Lambda le dimensioni del payload non possono superare i 6 MB. Per ulteriori informazioni sulle dimensioni dei batch Lambda, consulta «Comportamento in batch» nella sezione Lambda event source mappings nella Developer Guide. AWS Lambda

Compatibilità di Amazon DocumentDB con MongoDB

Amazon DocumentDB supporta la compatibilità con MongoDB, tra cui MongoDB 4.0, MongoDB 5.0 e MongoDB 8.0. La compatibilità con MongoDB significa che la maggior parte delle applicazioni, dei driver e degli strumenti che già utilizzi oggi con i tuoi database MongoDB possono essere utilizzati con Amazon DocumentDB con modifiche minime o nulle. Questa sezione descrive tutto ciò che devi sapere sulla compatibilità di Amazon DocumentDB con MongoDB, incluse nuove funzionalità e caratteristiche, guida introduttiva, percorsi di migrazione e differenze funzionali.

Argomenti

- Compatibilità con MongoDB 8.0
- Compatibilità con MongoDB 5.0
- Compatibilità con MongoDB 4.0
- Differenze funzionali: Amazon DocumentDB e MongoDB
- API, operazioni e tipi di dati MongoDB supportati in Amazon DocumentDB

Compatibilità con MongoDB 8.0

Argomenti

- Cosa c'è di nuovo in Amazon DocumentDB 8.0
- Inizia a usare Amazon DocumentDB 8.0
- Aggiornamento o migrazione ad Amazon DocumentDB 5.0 o 8.0
- Differenze funzionali

Cosa c'è di nuovo in Amazon DocumentDB 8.0

Amazon DocumentDB 8.0 migliora le prestazioni delle query fino a 7 volte e migliora il rapporto di compressione fino a 5 volte, consentendoti di creare applicazioni ad alte prestazioni a un costo inferiore. Amazon DocumentDB 8.0 è completamente compatibile con il protocollo cablato con MongoDB 8.0. Il riepilogo riportato di seguito presenta alcune delle principali funzionalità introdotte in Amazon DocumentDB 8.0. Per un elenco completo delle nuove funzionalità, consulta. Note di rilascio

- Fornisce la compatibilità con MongoDB 8.0 aggiungendo il supporto per i driver API MongoDB 8.0. Supporta anche le applicazioni create utilizzando le versioni dell'API MongoDB 6.0 e 7.0.

- Il nuovo pianificatore di query in Amazon DocumentDB 8.0 estende i miglioramenti delle prestazioni agli operatori della fase di aggregazione, oltre a supportare l'ottimizzazione della pipeline di aggregazione e comandi distinti.
- Amazon DocumentDB 8.0 supporta la raccolta e le visualizzazioni.
- Offre 6 nuove fasi di aggregazione: $replaceWith, $vectorSearch, $merge, $set, $unset, $bucket e 3 nuovi operatori di aggregazione $pow, $rand, $DateTrunc.
- Una nuova versione di text index: Text index v2 in Amazon DocumentDB 8.0 introduce token aggiuntivi, migliorando le funzionalità di ricerca testuale.
- Grazie alla creazione parallela di indici vettoriali, Amazon DocumentDB 8.0 riduce il tempo di creazione dell'indice fino a 30 volte.
- Limite di archiviazione aumentato a 256 TiB.

Inizia a usare Amazon DocumentDB 8.0

Per iniziare a usare Amazon DocumentDB 8.0, consulta il. Inizia a usare Amazon DocumentDB Puoi creare un nuovo cluster Amazon DocumentDB 8.0 utilizzando Console di gestione AWS o l' AWS SDK,, o. AWS CLI CloudFormation Quando ci si connette ad Amazon DocumentDB, è necessario utilizzare un driver o un'utilità MongoDB compatibile con MongoDB 5.0 o versioni successive.

> **Note**
>
> Quando si utilizza l' AWS SDK, o AWS CLI, la versione del motore sarà predefinita CloudFormation la 5.0.0. È necessario specificare esplicitamente il parametro `engineVersion = 8.0.0` per creare un nuovo cluster Amazon DocumentDB 8.0 o per creare un nuovo cluster Amazon DocumentDB 4.0 o `engineVersion = 4.0.0` per creare un nuovo cluster Amazon DocumentDB `engineVersion = 3.6.0` 3.6. Per un determinato cluster Amazon DocumentDB, puoi determinare la versione del cluster utilizzando la chiamata `describe-db-clusters` o la console di gestione Amazon DocumentDB AWS CLI per visualizzare il numero di versione del motore per un determinato cluster.

Amazon DocumentDB 5.0 e 8.0 supportano i processori Amazon EC2 `r8g` Graviton4 e Graviton2 come i tipi di `t4.medium` istanza per i cluster `r6g` ed è disponibile in tutti i modelli supportati (vedi). Regioni AWS Istanze Per ulteriori informazioni sui prezzi, consulta la sezione Prezzi di Amazon DocumentDB (con compatibilità MongoDB). https://aws.amazon.com/documentdb/pricing/

Aggiornamento o migrazione ad Amazon DocumentDB 5.0 o 8.0

Puoi migrare da MongoDB 3.6 o MongoDB 4.0 ad Amazon DocumentDB 5.0 o Amazon DocumentDB 8.0 utilizzando le utilità or come mongodump, mongorestore, mongoimport e mongoexport. AWS DMS ??? Per istruzioni Aggiornamento del cluster Amazon DocumentDB tramite AWS Database Migration Service sulla migrazione, consulta.

Differenze funzionali

Differenze funzionali tra Amazon DocumentDB 5.0 e 8.0

Con il rilascio di Amazon DocumentDB 8.0, ci sono differenze funzionali tra Amazon DocumentDB 5.0 e Amazon DocumentDB 8.0:

- Planner v1 è il pianificatore di query predefinito in Amazon DocumentDB 5.0, mentre il Planner v3, più performante, è l'impostazione predefinita in Amazon DocumentDB 8.0.
- Le nuove funzionalità di Amazon DocumentDB 8.0, tra cui Views, Collation e operatori come $merge, sono compatibili solo con Planner v3.
- La compressione è attivata per impostazione predefinita in Amazon DocumentDB 8.0 ed è impostata per utilizzare l'algoritmo Zstandard. Inoltre, in Amazon DocumentDB 8.0 «abilitato» non è più una scelta valida; puoi scegliere tra Zstd, LZ4 e nessuna.
- Amazon DocumentDB 8.0 supporta solo TLS 1.2 e versioni successive per le connessioni al database.

> **Important**
>
> Amazon DocumentDB 8.0 richiede TLS 1.2 o versioni successive. TLS 1.0 e TLS 1.1 non sono più supportati.

Compatibilità con MongoDB 5.0

Argomenti

- Cosa c'è di nuovo in Amazon DocumentDB 5.0
- Cosa c'è di nuovo in Amazon DocumentDB 5.0.1
- Inizia a usare Amazon DocumentDB 5.0

- Aggiornamento o migrazione ad Amazon DocumentDB 5.0
- Differenze funzionali

Cosa c'è di nuovo in Amazon DocumentDB 5.0

Amazon DocumentDB 5.0 introduce nuove caratteristiche e funzionalità che includono limiti di archiviazione e crittografia a livello di campo sul lato client. Il riepilogo riportato di seguito presenta alcune delle principali funzionalità introdotte in Amazon DocumentDB 5.0. Per un elenco completo delle nuove funzionalità, consulta. Note di rilascio

- Limite di archiviazione aumentato a 128 TiB per tutti i cluster Amazon DocumentDB basati su istanze e i cluster elastici basati su shard.
- Presentazione del motore Amazon DocumentDB 5.0 (versione 3.0.775)
  - Supporto per i driver API MongoDB 5.0
  - Supporto per la crittografia a livello di Client-side campo (FLE). Ora puoi crittografare i campi sul lato client prima di scrivere i dati nel cluster Amazon DocumentDB. Per ulteriori informazioni, consulta Client-side crittografia a livello di campo.
  - Nuovi operatori di aggregazione:, `$dateAdd` `$dateSubtract`
  - Supporti per indici con operatore. `$elemMatch` Di conseguenza, le interrogazioni `$elemMatch` comporteranno scansioni degli indici.

Amazon DocumentDB non supporta tutte le funzionalità di MongoDB 5.0. Quando abbiamo creato Amazon DocumentDB 5.0, abbiamo lavorato a ritroso sulla base delle funzionalità e delle funzionalità che i nostri clienti ci chiedevano di sviluppare maggiormente. Continueremo ad aggiungere ulteriori funzionalità di MongoDB 5.0 in base a ciò che i clienti ci chiedono di creare. Per l'elenco più recente delle API supportate, consulta. API, operazioni e tipi di dati MongoDB supportati in Amazon DocumentDB

Cosa c'è di nuovo in Amazon DocumentDB 5.0.1

Amazon DocumentDB 5.0.1 è la prima versione secondaria della versione principale di Amazon DocumentDB 5.0. Le versioni minori includono miglioramenti come nuove funzionalità, correzioni di sicurezza e correzioni di bug. È possibile eseguire l'aggiornamento alla versione 5.0.1 dalla versione 5.0.0 utilizzando un aggiornamento della versione secondaria. Per ulteriori informazioni, consulta Aggiornamento della versione secondaria di Amazon DocumentDB.

- Funzionalità di aggregazione migliorate con nuovi operatori (`$rand`,,,`$pow`). `$dateToParts` `$dateFromParts`
- Metrica delle connessioni attive per monitorare le istanze. Per ulteriori informazioni, consulta Quote delle istanze.
- Metriche granulari delle prestazioni a livello di comando CloudWatch per le operazioni CRUD (`find`,,,`insert`,,`findAndModify`,`update`,`delete`,`aggregate`,`commitTransaction`) `count` `distinct` `getMoreabortTransaction`, l'esposizione e le dimensioni per un migliore monitoraggio `AvgDuration` operativo. `P100Duration` `MaxConcurrent`
- Amazon DocumentDB 5.0.1 supporta solo TLS 1.2 e versioni successive per le connessioni al database.
- Sono state aggiunte `gcRuntimeStats` e suddivise le metriche relative al livello di raccolta `collstats` per fornire maggiori informazioni sullo stato generale del processo di raccolta dei rifiuti. `documentFragmentStats` `storageSegmentBase` `storageSegmentExtended`

Inizia a usare Amazon DocumentDB 5.0

Per iniziare a usare Amazon DocumentDB 5.0, consulta. Inizia a usare Amazon DocumentDB Puoi creare un nuovo cluster Amazon DocumentDB 5.0 utilizzando Console di gestione AWS o l' AWS SDK,, AWS CLI o. CloudFormation Quando ci si connette ad Amazon DocumentDB, è necessario utilizzare un driver o un'utilità MongoDB compatibile con MongoDB 5.0 o versioni successive.

> **Note**
>
> Quando si utilizza l' AWS SDK, o AWS CLI, la versione del motore sarà predefinita CloudFormation la 5.0.0. È necessario specificare esplicitamente il parametro `engineVersion = 4.0.0` per creare un nuovo cluster Amazon DocumentDB 4.0 o `engineVersion = 3.6.0` per creare un nuovo cluster Amazon DocumentDB 3.6. Per un determinato cluster Amazon DocumentDB, puoi determinare la versione del cluster utilizzando la chiamata `describe-db-clusters` o la console AWS CLI di gestione Amazon DocumentDB per visualizzare il numero di versione del motore per un determinato cluster.

Amazon DocumentDB 5.0 e 8.0 supportano i processori Amazon EC2 `r8g` Graviton4 e Graviton2 come i tipi di `t4.medium` istanza per i cluster `r6g` ed è disponibile in tutti i modelli supportati (vedi). Regioni AWS Istanze Per ulteriori informazioni sui prezzi, consulta la sezione Prezzi di Amazon DocumentDB (con compatibilità MongoDB). https://aws.amazon.com/documentdb/pricing/

Aggiornamento o migrazione ad Amazon DocumentDB 5.0

Puoi migrare da MongoDB 3.6 o MongoDB 4.0 ad Amazon DocumentDB 5.0 utilizzando le utilità or come mongodump, AWS DMS mongorestore, mongoimport e mongoexport. ??? Per istruzioni Aggiornamento del cluster Amazon DocumentDB tramite AWS Database Migration Service sulla migrazione, consulta.

Differenze funzionali

Differenze funzionali tra Amazon DocumentDB 4.0 e 5.0

Con il rilascio di Amazon DocumentDB 5.0, ci sono differenze funzionali tra Amazon DocumentDB 4.0 e Amazon DocumentDB 5.0:

- Il ruolo integrato di backup ora supporta. `serverStatus` Azione: gli sviluppatori e le applicazioni con ruolo di backup possono raccogliere statistiche sullo stato del cluster Amazon DocumentDB.
- Il `SecondaryDelaySecs` campo si sostituisce `slaveDelay` nell'`replSetGetConfig`output.
- Il hello comando replaces`isMaster`: hello restituisce un documento che descrive il ruolo di un cluster Amazon DocumentDB.
- Amazon DocumentDB 5.0 ora supporta le scansioni degli indici con l'`$elemMatch`operatore nel primo livello di nidificazione. Le scansioni degli indici sono supportate quando il solo filtro di interrogazione ha un livello di `$elemMatch` filtro, ma non sono supportate se è inclusa una query `$elemMatch` annidata.

  Ad esempio, in Amazon DocumentDB 5.0, se includi l'`$elemMatch`operatore nel livello annidato, non restituirà un valore come in Amazon DocumentDB 4.0:

```
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 ] } ] ] }
```

- La proiezione «$» in Amazon DocumentDB 4.0 restituisce tutti i documenti con tutti i campi. Con Amazon DocumentDB 5.0, il find comando con proiezione «$» restituisce i documenti che corrispondono al parametro di query contenenti solo il campo corrispondente alla proiezione «$».
- In Amazon DocumentDB 5.0, i find comandi con `$regex` e i parametri di `$options` query restituiscono un errore: «Impossibile impostare le opzioni in entrambi e». `$regex` `$options`
- Con Amazon DocumentDB 5.0, `$indexOfCP` ora restituisce «-1" quando:
  - la sottostringa non si trova nell'espressione della stringa, oppure
  - start è un numero maggiore di end, o
  - start è un numero maggiore della lunghezza in byte della stringa.
- In Amazon DocumentDB 4.0, `$indexOfCP` restituisce «0" quando la posizione iniziale è un numero maggiore di end o della lunghezza in byte della stringa.
- Con Amazon DocumentDB 5.0, le operazioni di proiezione, ad esempio `_id` `fields{"_id.nestedField" : 1}`, restituiscono documenti che includono solo il campo proiettato. Mentre in Amazon DocumentDB 4.0, i comandi di proiezione di campo annidati non filtrano alcun documento.

Compatibilità con MongoDB 4.0

Argomenti

- Caratteristiche di Amazon DocumentDB 4.0
- Inizia a usare Amazon DocumentDB 4.0
- Aggiornamento o migrazione ad Amazon DocumentDB 4.0
- Differenze funzionali

Caratteristiche di Amazon DocumentDB 4.0

Amazon DocumentDB 4.0 ha introdotto molte nuove caratteristiche e funzionalità che includono transazioni ACID e miglioramenti ai flussi di modifiche. Il riepilogo seguente mostra alcune delle principali funzionalità introdotte in Amazon DocumentDB 4.0. Per visualizzare un elenco completo delle funzionalità, consultaNote di rilascio.

- Transazioni ACID: Amazon DocumentDB ora supporta la capacità di eseguire transazioni su più documenti, rendiconti, raccolte e database. Le transazioni semplificano lo sviluppo di applicazioni consentendo di eseguire operazioni atomiche, coerenti, isolate e durevoli (ACID) su uno o più documenti all'interno di un cluster Amazon DocumentDB. Per ulteriori informazioni, consulta Transazioni in Amazon DocumentDB.
- Flussi di modifica: ora puoi aprire un flusso di modifiche a livello di cluster (`client.watch()omongo.watch()`) e di database (`db.watch()`), puoi specificare un `startAtOperationTime` per aprire un cursore del flusso di modifiche e infine puoi estendere il periodo di conservazione del flusso di modifiche a 7 giorni (prima 24 ore). Per ulteriori informazioni, consulta Utilizzo dei flussi di modifiche con Amazon DocumentDB.
- AWS Database Migration Service(AWS DMS): Ora puoi utilizzarlo AWS DMS per migrare i tuoi carichi di lavoro MongoDB 4.0 su Amazon DocumentDB. AWS DMS ora supporta una fonte MongoDB 4.0, una destinazione Amazon DocumentDB 4.0 e una fonte Amazon DocumentDB 3.6 per eseguire aggiornamenti tra Amazon DocumentDB 3.6 e 4.0. Per ulteriori informazioni, consulta la documentazione di AWS DMS.
- Prestazioni e indicizzazione: ora puoi utilizzare un indice con`$lookup`, trovare query con una proiezione che contengono un campo o un campo e il `_id` campo può essere servito direttamente dall'indice e senza bisogno di leggere dalla raccolta (query coperta), la possibilità di utilizzare `hint()` con`findAndModify`, ottimizzazioni delle prestazioni e miglioramenti per `$addToSet` ridurre le dimensioni complessive dell'indice. Per ulteriori informazioni, consulta Note di rilascio.
- Operatori: Amazon DocumentDB 4.0 ora supporta una serie di nuovi operatori di aggregazione:,,,,,. `$ifNull` `$replaceRoot` `$setIsSubset` `$setIntersection` `$setUnion` `$setEquals` Puoi vedere tutte le API, le operazioni e i tipi di dati MongoDB che supportiamo su. API, operazioni e tipi di dati MongoDB supportati in Amazon DocumentDB
- Controllo degli accessi basato sui ruoli (RBAC): con entrambi `ListDatabase` i comandi `ListCollection` e ora puoi opzionalmente utilizzare i `authorizedDatabases` parametri `authorizedCollections` and per consentire agli utenti di elencare le raccolte e i database a cui hanno il permesso di accedere senza richiedere rispettivamente i ruoli e. `listCollections`

`listDatabase` Hai anche la possibilità di eliminare i tuoi cursori senza richiedere il ruolo. `KillCursor`

Amazon DocumentDB non supporta tutte le funzionalità di MongoDB 4.0. Quando abbiamo creato Amazon DocumentDB 4.0, abbiamo lavorato a ritroso sulla base delle funzionalità e delle funzionalità che i nostri clienti ci chiedevano di sviluppare maggiormente. Continueremo ad aggiungere ulteriori funzionalità di MongoDB 4.0 in base a ciò che i clienti ci chiedono di creare. Ad esempio, Amazon DocumentDB 4.0 attualmente non supporta gli operatori di conversione dei tipi o gli operatori di stringa introdotti in MongoDB 4.0. Per l'elenco più recente delle API supportate, consulta. API, operazioni e tipi di dati MongoDB supportati in Amazon DocumentDB

Inizia a usare Amazon DocumentDB 4.0

Per iniziare a usare Amazon DocumentDB 4.0, consulta. Inizia a usare Amazon DocumentDB Puoi creare un nuovo cluster Amazon DocumentDB 4.0 utilizzando Console di gestione AWS o l' AWS SDK,, AWS CLI o. CloudFormation Quando ci si connette ad Amazon DocumentDB, è necessario utilizzare un driver o un'utilità MongoDB compatibile con MongoDB 4.0 o versioni successive.

> **Note**
>
> Quando si utilizza l' AWS SDK, o AWS CLI, la versione del motore sarà predefinita CloudFormation la 5.0.0. È necessario specificare esplicitamente il parametro `engineVersion = 4.0.0` per creare un nuovo cluster Amazon DocumentDB 4.0 o `engineVersion = 3.6.0` per creare un nuovo cluster Amazon DocumentDB 3.6. Per un determinato cluster Amazon DocumentDB, puoi determinare la versione del cluster utilizzando la chiamata `describe-db-clusters` o la console AWS CLI di gestione Amazon DocumentDB per visualizzare il numero di versione del motore per un determinato cluster.

Amazon DocumentDB 4.0 supportar5, `r6gt3.medium`, e tipi di `t4g.medium` istanza per i tuoi cluster ed è disponibile in tutti quelli supportati. Regioni AWS Non sono previsti costi aggiuntivi per l'utilizzo di Amazon DocumentDB 4.0. Per ulteriori informazioni sui prezzi, consulta la sezione Prezzi di Amazon DocumentDB (con compatibilità MongoDB).

Aggiornamento o migrazione ad Amazon DocumentDB 4.0

Puoi migrare da MongoDB 3.6 o MongoDB 4.0 ad Amazon DocumentDB 4.0 utilizzando le utilità or come mongodump, AWS DMS mongorestore, mongoimport e mongoexport. ??? Allo stesso

modo, puoi utilizzare gli stessi strumenti per eseguire l'aggiornamento da Amazon DocumentDB 3.6 ad Amazon DocumentDB 4.0. Per istruzioni sulla migrazione, consulta. Aggiornamento del cluster Amazon DocumentDB tramite AWS Database Migration Service

Differenze funzionali

Differenze funzionali tra Amazon DocumentDB 3.6 e 4.0

Con il rilascio di Amazon DocumentDB 4.0, ci sono differenze funzionali tra Amazon DocumentDB 3.6 e Amazon DocumentDB 4.0:

- Proiezione per documenti annidati: Amazon DocumentDB 3.6 considera il primo campo di un documento nidificato quando applica una proiezione. Tuttavia, Amazon DocumentDB 4.0 analizzerà i documenti secondari e applicherà la proiezione anche a ciascun documento secondario. Ad esempio: se la proiezione è`"a.b.c": 1`, il comportamento in entrambe le versioni è identico. Tuttavia, se la proiezione è`{a:{b:{c:1}}}`, Amazon DocumentDB 3.6 applicherà la proiezione solo a «a» e non a «b» o «c».
- Comportamento per **`minKey maxKey`**: in Amazon DocumentDB 4.0, il comportamento per non `{x:{$gt:MaxKey}}` restituisce nulla e per restituisce tutto. `{x:{$lt:MaxKey}}`
- Differenze tra documenti: il confronto di valori numerici di diversi tipi (double, int, long) nei documenti secondari (ad esempio, b in`{"_id" :1, "a" :{"b":1}}`) ora fornisce un risultato coerente tra i tipi di dati numerici e per ogni livello di un documento.

Differenze funzionali tra Amazon DocumentDB 4.0 e MongoDB 4.0

Di seguito sono riportate le differenze funzionali tra Amazon DocumentDB 4.0 e MongoDB 4.0.

- Ricerca con chiave vuota nel percorso: quando una raccolta contiene un documento con una chiave vuota all'interno dell'array (ad esempio`{"x" : [ { "" : 10 }, { "b" : 20 } ]}`) e quando la chiave utilizzata nella query termina con una stringa vuota (ad esempio`x.`), Amazon DocumentDB restituirà quel documento poiché attraversa tutti i documenti dell'array mentre MongoDB non restituirà quel documento.
- **`$setOnInsert`**insieme **`$`** nel percorso: l'operatore sul campo non `$setOnInsert` funzionerà in combinazione con nel percorso $ in Amazon DocumentDB, che è anche coerente con MongoDB 4.0.

Differenze funzionali: Amazon DocumentDB e MongoDB

Di seguito sono riportate le differenze funzionali tra Amazon DocumentDB (con compatibilità MongoDB) e MongoDB.

Argomenti

- Vantaggi funzionali di Amazon DocumentDB
- Differenze funzionali aggiornate
- Differenze funzionali con MongoDB

Vantaggi funzionali di Amazon DocumentDB

Transazioni implicite

In Amazon DocumentDB, tutte le istruzioni CRUD (`findAndModify`,, `updateinsert`,`delete`) garantiscono atomicità e coerenza, anche per le operazioni che modificano più documenti. Con il lancio di Amazon DocumentDB 4.0, sono ora supportate le transazioni esplicite che forniscono proprietà ACID per operazioni con più dichiarazioni e raccolte multiple. Per ulteriori informazioni sull'utilizzo delle transazioni in Amazon DocumentDB, consulta. Transazioni in Amazon DocumentDB

Di seguito sono riportati alcuni esempi di operazioni in Amazon DocumentDB che modificano più documenti che soddisfano comportamenti sia atomici che coerenti.

```
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 }]
})
```

Le singole operazioni che compongono operazioni di massa come `updateMany` e `deleteMany` sono atomiche, ma la totalità dell'operazione di massa non è atomica. Ad esempio, la totalità dell'operazione `insertMany` è atomica se le singole operazioni di inserimento vengono eseguite correttamente senza errori. Se si verifica un errore con un'operazione `insertMany`, ogni singola istruzione insert all'interno dell'operazione `insertMany` verrà eseguita come operazione atomica. Se sono necessarie proprietà ACID per `insertMany` e `deleteMany` operazioni, si consiglia di utilizzare una transazione. `updateMany`

Differenze funzionali aggiornate

Amazon DocumentDB continua a migliorare la compatibilità con MongoDB sfruttando le funzionalità che i nostri clienti ci chiedono di creare. Questa sezione contiene le differenze funzionali che abbiamo rimosso in Amazon DocumentDB per semplificare le migrazioni e la creazione di applicazioni per i nostri clienti.

Argomenti

- Indicizzazione degli array

- Multi-key indici
- Caratteri nulli nelle stringhe
- Role-based controllo degli accessi
- indicizzazione $regex
- Proiezione per documenti annidati

Indicizzazione degli array

A partire dal 23 aprile 2020, Amazon DocumentDB supporta ora la possibilità di indicizzare array di dimensioni superiori a 2.048 byte. Il limite per un singolo elemento in un array rimane di 2.048 byte, il che è coerente con MongoDB.

Se si sta creando un nuovo indice, non è necessaria alcuna operazione per sfruttare le funzionalità migliorate. Se si dispone di un indice esistente, è possibile sfruttare le funzionalità migliorate rilasciando l'indice e quindi ricreandolo. La versione dell'indice corrente con le funzionalità migliorate è `"v" : 3`.

> **Note**
>
> Per i cluster di produzione, l'eliminazione dell'indice potrebbe avere un impatto sulle prestazioni dell'applicazione. Effettua prima i test e procedi con cautela quando apporti modifiche a un sistema di produzione. Inoltre, il tempo necessario per ricreare l'indice dipende dalla dimensione complessiva dei dati della raccolta.

È possibile eseguire una query per la versione degli indici utilizzando il seguente comando.

```
db.collection.getIndexes()
```

L'aspetto dell'output di questa operazione è simile al seguente. In questo output, la versione dell'indice è `"v" : 3`, che è quella più recente.

```
[
    {
        "v" : 3,
        "key" : {
        "_id" : 1
```

```
        },
        "name" : "_id_",
        "ns" : "test.test"
    }
]
```

Multi-key indici

A partire dal 23 aprile 2020, Amazon DocumentDB supporta ora la possibilità di creare un indice composto con più chiavi nello stesso array.

Se si sta creando un nuovo indice, non è necessaria alcuna operazione per sfruttare le funzionalità migliorate. Se si dispone di un indice esistente, è possibile sfruttare le funzionalità migliorate rilasciando l'indice e quindi ricreandolo. La versione dell'indice corrente con le funzionalità migliorate è `"v" : 3`.

> **Note**
>
> Per i cluster di produzione, l'eliminazione dell'indice potrebbe avere un impatto sulle prestazioni dell'applicazione. Effettua prima i test e procedi con cautela quando apporti modifiche a un sistema di produzione. Inoltre, il tempo necessario per ricreare l'indice dipende dalla dimensione complessiva dei dati della raccolta.

È possibile eseguire una query per la versione degli indici utilizzando il seguente comando.

```
db.collection.getIndexes()
```

L'aspetto dell'output di questa operazione è simile al seguente. In questo output, la versione dell'indice è `"v" : 3`, che è quella più recente.

```
[
    {
        "v" : 3,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.test"
    }
```

```
]
```

Caratteri nulli nelle stringhe

A partire dal 22 giugno 2020, Amazon DocumentDB ora supporta i caratteri nulli () `'\0'` nelle stringhe.

Role-based controllo degli accessi

A partire dal 26 marzo 2020, Amazon DocumentDB supporta il controllo degli accessi basato sui ruoli (RBAC) per i ruoli integrati. Per ulteriori informazioni, consulta Role-Based Controllo degli accessi.

`indicizzazione $regex`

A partire dal 22 giugno 2020, Amazon DocumentDB ora supporta la possibilità per `$regex` gli operatori di utilizzare un indice.

Per utilizzare un indice con l'operatore `$regex`, è necessario utilizzare il comando `hint()`. Quando si utilizza `hint()`, è necessario specificare il nome del campo in cui si sta applicando `$regex`. Ad esempio, se si dispone di un indice nel campo `product` con il nome dell'indice come `p_1`, `db.foo.find({product: /^x.*/}).hint({product:1})` utilizzerà l'indice `p_1`, ma `db.foo.find({product: /^x.*/}).hint("p_1")` non utilizzerà l'indice. È possibile verificare se viene scelto un indice utilizzando il comando `explain()` o utilizzando il profiler per la registrazione di query lente. Ad esempio, `db.foo.find({product: /^x.*/}).hint("p_1").explain()`.

> **Note**
>
> Il metodo `hint()` può essere utilizzato solo con un indice alla volta.

L'utilizzo di un indice per una query `$regex` è ottimizzato per le query regex che utilizzano un prefisso e non specificano le opzioni regex `i`, `m` o `o`.

Quando si utilizza un indice con `$regex`, si consiglia di creare un indice in campi altamente selettivi in cui il numero di valori duplicati è inferiore all'1% del numero totale di documenti nella raccolta. Ad esempio, se la raccolta contiene 100.000 documenti, creare solo indici nei campi in cui lo stesso valore si verifica al massimo 1000 volte.

Proiezione per documenti annidati

Esiste una differenza funzionale con `$project` l'operatore tra Amazon DocumentDB e MongoDB nella versione 3.6 che è stata risolta in Amazon DocumentDB 4.0 ma rimarrà non supportata in Amazon DocumentDB 3.6.

Amazon DocumentDB 3.6 considera solo il primo campo in un documento annidato quando applica una proiezione, mentre MongoDB 3.6 analizzerà i documenti secondari e applicherà la proiezione anche a ciascun documento secondario.

Ad esempio: se la proiezione è, il comportamento funziona come previsto sia in Amazon DocumentDB che in `“a.b.c”: 1` MongoDB. Tuttavia, se la proiezione è`{a:{b:{c:1}}}`, Amazon DocumentDB 3.6 applicherà la proiezione solo a e non a or. `a b c` In Amazon DocumentDB 4.0, la proiezione `{a:{b:{c:1}}}` verrà applicata a, e. `a b c`

Differenze funzionali con MongoDB

Argomenti

- Operatore $vectorSearch
- OpCountersCommand
- Database e raccolte di amministrazione
- CursorMaxTimeMS
- explain()
- Costruzioni di indici
- Ricerca con chiave vuota nel percorso
- API, operazioni e tipi di dati MongoDB
- Utilità mongodump e mongorestore
- Ordinamento dei risultati
- Retryable scrive
- Indice Sparse
- Utilizzo di $elemMatch all'interno di un'espressione $all
- Dollaro ($) e punto (.) nei nomi dei campi
- $lookup

- $natural e ordinamento inverso

Operatore $vectorSearch

Amazon DocumentDB non supporta `$vectorSearch` come operatore indipendente. Supportiamo invece, `vectorSearch` all'interno dell'`$search`operatore. Per ulteriori informazioni, consulta Ricerca vettoriale per Amazon DocumentDB.

OpCountersCommand

Il `OpCountersCommand` comportamento di Amazon DocumentDB si discosta da quello di MongoDB come segue: `opcounters.command`

- MongoDB `opcounters.command` conta tutti i comandi tranne insert, update ed delete, mentre Amazon DocumentDB esclude anche il comando. `OpCountersCommand` `find`
- Amazon DocumentDB conta alcuni comandi interni verso. `OpCountersCommand`

Database e raccolte di amministrazione

Amazon DocumentDB non supporta rispettivamente il database di amministrazione o locale né MongoDB `system.*` o `startup_log` le raccolte.

CursorMaxTimeMS

In Amazon DocumentDB, `cursor.maxTimeMS` reimposta il contatore per ogni richiesta. `getMore` Pertanto, se `maxTimeMS` viene specificato un valore di 3000 MS, la query richiede 2800 MS e ogni `getMore` richiesta successiva richiede 300 MS, il cursore non scadrà. Il cursore scade solo quando una singola operazione, la query o una singola richiesta, richiede più di quanto specificato. `getMore` `maxTimeMS` Inoltre, lo sweeper che controlla il tempo di esecuzione del cursore viene eseguito con una granularità di cinque (5) minuti.

explain()

Amazon DocumentDB emula le API MongoDB 3.6, 4.0, 5.0 e 8.0 su un motore di database appositamente progettato che utilizza un sistema di storage distribuito, tollerante ai guasti e autoriparante. Di conseguenza, i piani di interrogazione e l'output di possono differire tra Amazon DocumentDB e MongoDB. `explain()` I clienti che desiderano il controllo sul piano di query possono utilizzare l'operatore `$hint` per applicare la selezione di un indice preferito.

Costruzioni di indici

Amazon DocumentDB consente la creazione di un solo indice su una raccolta alla volta. In primo piano o sullo sfondo. Se operazioni come `createIndex()` o `dropIndex()` si verificano nella stessa raccolta quando è in corso una creazione di indice, l'operazione appena tentata avrà esito negativo.

Per impostazione predefinita, le compilazioni degli indici in Amazon DocumentDB e MongoDB versione 4.0 vengono visualizzate in primo piano. MongoDB versione 4.2 e successive ignora l'opzione di creazione dell'indice in background se specificata su createIndexes o sui relativi helper di shell e. `createIndex()` `createIndexes()`

Un indice Time to Live (TTL) inizia a far scadere i documenti dopo il completamento della creazione dell'indice.

Ricerca con chiave vuota nel percorso

Quando esegui la ricerca con una chiave che include una stringa vuota come parte del percorso (ad esempio `x.x..b`) e l'oggetto ha un percorso chiave di stringa vuoto (ad esempio`{"x" : [ { "" : 10 }, { "b" : 20 } ]}`) all'interno di un array, Amazon DocumentDB restituirà risultati diversi rispetto a quelli ottenuti se eseguissi la stessa ricerca in MongoDB.

In MongoDB, la ricerca di un percorso chiave vuoto all'interno dell'array funziona come previsto quando la chiave di stringa vuota non si trova alla fine della ricerca del percorso. Tuttavia, quando la chiave di stringa vuota si trova alla fine della ricerca del percorso, non esamina l'array.

Tuttavia, in Amazon DocumentDB, viene letto solo il primo elemento all'interno dell'array, perché `getArrayIndexFromKeyString` converte una stringa vuota in`0`, quindi la ricerca della chiave di stringa viene considerata come ricerca dell'indice dell'array.

API, operazioni e tipi di dati MongoDB

Amazon DocumentDB è compatibile con le API MongoDB 3.6, 4.0, 5.0 e 8.0. Per l'elenco aggiornato delle funzionalità supportate, vedi API, operazioni e tipi di dati MongoDB supportati in Amazon DocumentDB.

Utilità mongodump e mongorestore

Amazon DocumentDB non supporta un database di amministrazione e pertanto non esegue il dump o il ripristino del database di amministrazione quando si utilizzano le utilità or.

`mongodump` `mongorestore` Quando si crea un nuovo database in Amazon DocumentDB utilizzando`mongorestore`, è necessario ricreare i ruoli utente oltre all'operazione di ripristino.

> **Requisiti di versione di MongoDB Database Tools**
>
> Usa MongoDB Database Tools fino alla versione 100.11.0 inclusa per Amazon DocumentDB. Per scaricarlo, consulta le versioni di MongoDB Database Tools sul sito Web di MongoDB.

Ordinamento dei risultati

Amazon DocumentDB non garantisce l'ordinamento implicito dei set di risultati. Per garantire l'ordinamento di un set di risultati, specifica in modo esplicito un ordinamento utilizzando `sort()`.

L'esempio seguente ordina gli elementi della raccolta dell'inventario in ordine decrescente in base al campo del magazzino.

```
db.inventory.find().sort({ stock: -1 })
```

Quando si utilizza la fase di `$sort` aggregazione, l'ordinamento non viene mantenuto a meno che la `$sort` fase non sia l'ultima fase della pipeline di aggregazione. Quando si utilizza la fase di `$sort` aggregazione in combinazione con la fase di `$group` aggregazione, la fase di `$sort` aggregazione viene applicata solo agli accumulatori e. `$first` `$last` In Amazon DocumentDB 4.0, è stato aggiunto il supporto per `$push` rispettare l'ordinamento della fase precedente. `$sort`

Retryable scrive

La maggior parte dei MongoDB-compatible driver attuali abilita le scritture riutilizzabili per impostazione predefinita. Amazon DocumentDB supporta le scritture riutilizzabili a partire dalla versione del motore 8.0.2. Nelle versioni precedenti del motore, le scritture riutilizzabili non sono supportate ed è necessario disabilitarle per evitare errori.

Nelle versioni del motore precedenti alla 8.0.2, l'attivazione delle scritture riutilizzabili genera un messaggio di errore simile al seguente.

```
{"ok":0,"errmsg":"Retryable writes are not supported","code":301,"name":"MongoError"}
```

Le scritture retryable possono essere disabilitate tramite la stringa di connessione (ad esempio`MongoClient("mongodb://my.mongodb.cluster/db?retryWrites=false")`)

o l'argomento della parola chiave del MongoClient costruttore (ad esempio,). `MongoClient("mongodb://my.mongodb.cluster/db", retryWrites=False)`

Per ulteriori informazioni sulle scritture retryable sulla versione del motore 8.0.2 e successive, vedere. Retryable scrive in Amazon DocumentDB

Indice Sparse

Per utilizzare un indice Sparse creato in una query, devi utilizzare la clausola `$exists` nei campi che coprono l'indice. Se ometti`$exists`, Amazon DocumentDB non utilizzerà l'indice sparse.

Di seguito è riportato un esempio di :

```
db.inventory.count({ "stock": { $exists: true }})
```

Per gli indici sparsi a più chiavi, Amazon DocumentDB non supporta un vincolo di chiave univoco se la ricerca di un documento dà come risultato un set di valori e manca solo un sottoinsieme dei campi indicizzati. Ad esempio, `createIndex({"a.b" : 1 }, { unique : true, sparse :true })` non è supportato, dato che l'input di `"a" : [ { "b" : 2 }, { "c" : 1 } ]`, come `"a.c"` è memorizzato nell'indice.

Utilizzo di $elemMatch all'interno di un'espressione $all

Amazon DocumentDB attualmente non supporta l'uso dell'`$elemMatch`operatore all'interno di un'espressione. `$all` Come soluzione alternativa, è possibile utilizzare l'operatore `$and` con `$elemMatch` come segue.

Funzionamento originale:

```
db.col.find({
  qty: {
    $all: [
      { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } },
      { "$elemMatch": { num: 40, size: "XL" } }
    ]
  }
})
```

Funzionamento aggiornato:

```
db.col.find({
  $and: [
    { qty: { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } } },
    { qty: { "$elemMatch": { qty: 40, size: "XL" } } }
  ]
})
```

Dollaro ($) e punto (.) nei nomi dei campi

Amazon DocumentDB non supporta l'interrogazione dei campi con prefisso Dollar ($) in $in, $nin e $all negli oggetti annidati. Ad esempio, la seguente query non è valida in Amazon DocumentDB:

```
coll.find({"field": {"$all": [{ "$a": 1 }]}})
```

`$lookup`

Amazon DocumentDB supporta la capacità di eseguire corrispondenze di uguaglianza (ad esempio, left outer join) e supporta anche sottoquery non correlate. A partire dalla versione 8.0.2, Amazon DocumentDB supporta anche le sottoquery correlate, inclusa la sintassi concisa delle sottoquery correlate. In una sottoquery correlata, solo un sottoinsieme di fasi di aggregazione che valutano le espressioni può fare riferimento alle variabili. `let` Per l'elenco delle fasi supportate e delle relative restrizioni, vedere. Fasi supportate nelle sottoquery correlate Nelle versioni precedenti alla 8.0.2, le sottoquery correlate non sono supportate.

Utilizzo di un indice con $lookup

È ora possibile utilizzare un indice con l'operatore stage. `$lookup` In base al caso d'uso, sono disponibili diversi algoritmi di indicizzazione che è possibile utilizzare per ottimizzare le prestazioni. Questa sezione spiegherà i diversi algoritmi di indicizzazione `$lookup` e ti aiuterà a scegliere quello migliore per il tuo carico di lavoro.

Per impostazione predefinita, Amazon DocumentDB utilizza l'algoritmo hash quando viene utilizzato e sort merge quando `allowDiskUse:false` viene utilizzato. `allowDiskUse:true`

> **Note**
>
> L'`allowDiskUse`opzione non è attualmente supportata per il comando. `find` L'opzione è supportata solo come parte dell'aggregazione. Usa il framework di aggregazione con

`allowDiskUse:true` per gestire query di grandi dimensioni che potrebbero superare i limiti di memoria.

In alcuni casi d'uso, potrebbe essere opportuno forzare l'ottimizzatore di query a utilizzare un algoritmo diverso. Di seguito sono riportati i diversi algoritmi di indicizzazione che l'operatore di `$lookup` aggregazione può utilizzare:

- Ciclo annidato: un piano di ciclo annidato è in genere utile per un carico di lavoro se la raccolta esterna è <1 GB e il campo della raccolta esterna ha un indice. Se viene utilizzato l'algoritmo del ciclo annidato, il piano di spiegazione mostrerà lo stage come. `NESTED_LOOP_LOOKUP`
- Sort merge: un piano di ordinamento unificato è in genere utile per un carico di lavoro se la raccolta esterna non ha un indice nel campo utilizzato nella ricerca e il set di dati di lavoro non è contenuto nella memoria. Se viene utilizzato l'algoritmo di ordinamento unito, il piano di spiegazione mostrerà la fase come. `SORT_LOOKUP`
- Hash: un piano hash è in genere utile per un carico di lavoro se la raccolta esterna è < 1 GB e il set di dati di lavoro è contenuto nella memoria. Se viene utilizzato l'algoritmo hash, il piano di spiegazione mostrerà la fase come. `HASH_LOOKUP`

È possibile identificare l'algoritmo di indicizzazione utilizzato per l'`$lookup`operatore utilizzando `explain` sulla query. Di seguito è riportato un esempio:

```
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
}
```

In alternativa all'utilizzo del `explain()` metodo, è possibile utilizzare il profiler per esaminare l'algoritmo utilizzato con l'utilizzo dell'`$lookup`operatore. Per ulteriori informazioni sul profiler, vedere. Profilazione delle operazioni di Amazon DocumentDB

Utilizzo di un **PlanHint**

Se desideri forzare l'ottimizzatore di query a utilizzare un algoritmo di indicizzazione diverso con`$lookup`, puoi utilizzare un. `planHint` Per farlo, usa le opzioni del commento nella fase di aggregazione per forzare un piano diverso. Di seguito è riportato un esempio della sintassi del commento:

```
comment : {
    comment :  "<string>",
    lookupStage : { planHint : "SORT" | "HASH" | "NESTED_LOOP" }
}
```

Di seguito è riportato un esempio di utilizzo di `planHint` per forzare l'ottimizzatore di query a utilizzare l'algoritmo di HASH indicizzazione:

```
db.foo.aggregate(
   [
      {
         $lookup:
            {
               from: "foo",
               localField: "_id",
               foreignField: "_id",
               as: "joined"
            },
      }
   ]
),
{
   comment : "{ \"lookupStage\" : { \"planHint\": \"HASH\" }}"
```

Per verificare quale algoritmo è più adatto al carico di lavoro, puoi utilizzare il `executionStats` parametro del `explain` metodo per misurare il tempo di esecuzione della `$lookup` fase modificando al contempo l'algoritmo di indicizzazione (ad esempio,//). HASH SORT NESTED_LOOP

L'esempio seguente mostra come utilizzare l'algoritmo `executionStats` per misurare il tempo di esecuzione della `$lookup` fase. SORT

```
db.foo.explain("executionStats").aggregate(
   [
      {
         $lookup:
            {
               from: "foo",
               localField: "_id",
               foreignField: "_id",
               as: "joined"
            },
      }
   ]
),
{
   comment : "{ \"lookupStage\" : { \"planHint\": \"SORT\" }}"
```

$natural e ordinamento inverso

Amazon DocumentDB supporta solo `$natural` le scansioni dirette delle raccolte. Le scansioni di raccolta inversa (`{$natural: -1}`) porteranno a un. `MongoServerError`

API, operazioni e tipi di dati MongoDB supportati in Amazon DocumentDB

Amazon DocumentDB (con compatibilità con MongoDB) è un servizio di database di documenti veloce, scalabile, altamente disponibile e completamente gestito che supporta i carichi di lavoro MongoDB. Amazon DocumentDB è compatibile con le API MongoDB 3.6, 4.0, 5.0 e 8.0. In questa sezione sono elencate le funzionalità supportate. Per assistenza nell'uso delle API e dei driver MongoDB, consulta i forum della community di MongoDB. Per ricevere assistenza tramite il servizio Amazon DocumentDB, contatta il team di supporto appropriato. AWS Per le differenze funzionali tra Amazon DocumentDB e MongoDB, consulta. Differenze funzionali: Amazon DocumentDB e MongoDB

Gli operatori e i comandi MongoDB solo interni o non applicabili a un servizio completamente gestito non sono supportati e non sono inclusi nell'elenco delle funzionalità supportate.

Abbiamo aggiunto oltre 50 funzionalità aggiuntive dal lancio e continueremo a lavorare procedendo a ritroso dai nostri clienti per fornire le funzionalità necessarie. Per informazioni sui lanci più recenti, consulta Amazon DocumentDB Announcements.

Se desideri che creiamo una funzionalità non supportata, faccelo sapere inviando un'email con il tuo ID account, le funzionalità richieste e il caso d'uso al team di assistenza di Amazon DocumentDB. mailto:documentdb-feature-request@amazon.com

Argomenti

- Comandi del database
- Operatori di interrogazione e proiezione
- Operatori di aggiornamento
- Dati geospaziali
- Metodi Cursor
- Operatori di pipeline di aggregazione
- Tipi di dati
- Indici

Comandi del database

Argomenti

- Comandi amministrativi
- Aggregazione
- Autenticazione
- Comandi diagnostici
- Operazioni di interrogazione e scrittura
- Comandi di gestione dei ruoli
- Comandi delle sessioni
- Gestione degli utenti
- Comandi di sharding

Comandi amministrativi

Comando	3.6	4.0	5.0	8.0	Cluster elastico
Capped Collections	No	No	No	No	No
clone CollectionAsCapped	No	No	No	No	No
collMod	Parziale	Parziale	Parziale	Parziale	Parziale
CollMod: scadeAfterSeconds	Sì	Sì	Sì	Sì	Sì
CollMod: nascosto	No	No	Sì (5.0.2+)	Sì (8.0.1+)	No
convertireToCapped	No	No	No	No	No
copydb	No	No	No	No	No
creazione	Sì	Sì	Sì	Sì	Sì
createView	No	No	No	Sì	No
createIndexes	Sì	Sì	Sì	Sì	Sì
currentOp	Sì	Sì	Sì	Sì	Sì
drop	Sì	Sì	Sì	Sì	Sì
dropDatabase	Sì	Sì	Sì	Sì	Sì

Comando	3.6	4.0	5.0	8.0	Cluster elastico
dropIndexes	Sì	Sì	Sì	Sì	Sì
filemd5	No	No	No	No	No
ottenere AuditConfig	No	Sì	Sì	Sì	No
killCursors	Sì	Sì	Sì	Sì	Sì
killOp	Sì	Sì	Sì	Sì	Sì
Elenca le collezioni*	Sì	Sì	Sì	Sì	Sì
listDatabases	Sì	Sì	Sì	Sì	Sì
listIndexes	Sì	Sì	Sì	Sì	Sì
reIndex	No	No	Sì	Sì	No
renameCollection	Sì	Sì	Sì	Sì	No
set AuditConfig	No	Sì	Sì	Sì	No

* La `type` chiave nell'opzione di filtro non è supportata.

Aggregazione

Comando	3.6	4.0	5.0	8.0	Cluster elastico
aggregate	Sì	Sì	Sì	Sì	Sì
count	Sì	Sì	Sì	Sì	Sì

Comando	3.6	4.0	5.0	8.0	Cluster elastico
distinct	Sì	Sì	Sì	Sì	Sì
mapReduce	No	No	No	Sì	No

Autenticazione

Comando	3.6	4.0	5.0	8.0	Cluster elastico
authenticate	Sì	Sì	Sì	Sì	Sì
Logout	Sì	Sì	Sì	Sì	Sì

Comandi diagnostici

Comando	3.6	4.0	5.0	8.0	Cluster elastico
buildInfo	Sì	Sì	Sì	Sì	Sì
collStats	Sì	Sì	Sì	Sì	Sì
conn PoolStats	No	No	No	No	No
connectio nStatus	Sì	Sì	Sì	Sì	Sì
dataSize	Sì	Sì	Sì	Sì	Sì
dbHash	No	No	No	No	No
dbStats	Sì	Sì	Sì	Sì	Sì

Comando	3.6	4.0	5.0	8.0	Cluster elastico
explain	Sì	Sì	Sì	Sì	Sì
explain: executionStats	Sì	Sì	Sì	Sì	Sì
caratteristiche	No	No	No	No	No
hostInfo	Sì	Sì	Sì	Sì	Sì
listCommands	Sì	Sì	Sì	Sì	Sì
pianoCacheClearFilters	No	No	Sì	Sì	No
pianoCacheListFilters	No	No	Sì	Sì	No
pianoCacheSetFilter	No	No	Sì	Sì	No
profiler	Sì	Sì	Sì	Sì	No
serverStatus	Sì	Sì	Sì	Sì	Sì
top	Sì	Sì	Sì	Sì	Sì

Operazioni di interrogazione e scrittura

Comando	3.6	4.0	5.0	8.0	Cluster elastico
Change streams	Sì	Sì	Sì	Sì	No
Elimina	Sì	Sì	Sì	Sì	Sì
find	Sì	Sì	Sì	Sì	Sì
trova AndModify	Sì	Sì	Sì	Sì	Sì
ottenere LastError	No	No	No	No	No
getMore	Sì	Sì	Sì	Sì	Sì
ottenere PrevError	No	No	No	No	No
GridFS	Sì	Sì	Sì	Sì	No
insert	Sì	Sì	Sì	Sì	Sì
parallelo Collectio nScan	No	No	No	No	No
resetError	No	No	No	No	No
aggiorna	Sì	Sì	Sì	Sì	Sì
ReplaceOne	Sì	Sì	Sì	Sì	Sì

Comandi di gestione dei ruoli

C	3.6	4.0	5.0	8.0 Cluster elastici
c e	Sì	Sì	Sì	Sì	No
g A r s	Sì	Sì	Sì	Sì	No
d	Sì	Sì	Sì	Sì	No
c F l	Sì	Sì	Sì	Sì	No
r F F	Sì	Sì	Sì	Sì	No
r F s	Sì	Sì	Sì	Sì	No
r	Sì	Sì	Sì	Sì	No
u e	Sì	Sì	Sì	Sì	No

Comandi delle sessioni

Comando	3.6	4.0	5.0	8.0	Cluster elastico
Transazione annullata	No	Sì	Sì	Sì	No
commitTransaction	No	Sì	Sì	Sì	No
Termina sessioni	No	No	No	No	No
uccidere AllSessions	No	Sì	Sì	Sì	No
uccidere AllSessionsByPattern	No	No	No	No	No
Uccidi sessioni	No	Sì	Sì	Sì	No
Aggiorna sessioni	No	No	No	No	No
Avvia sessione	No	Sì	Sì	Sì	No

Gestione degli utenti

Comando	3.6	4.0	5.0	8.0	Cluster elastico
createUser	Sì	Sì	Sì	Sì	Sì

Comando	3.6	4.0	5.0	8.0	Cluster elastico
goccia AllUsersF romDatabase	Sì	Sì	Sì	Sì	Sì
dropUser	Sì	Sì	Sì	Sì	Sì
concedere RolesToUser	Sì	Sì	Sì	Sì	Sì
revocare RolesFrom User	Sì	Sì	Sì	Sì	Sì
updateUser	Sì	Sì	Sì	Sì	Sì
Informazioni sugli utenti	Sì	Sì	Sì	Sì	Sì

Comandi di sharding

Comando	Cluster elastico
abortire ReshardCollection	No
aggiunge Shard	No
aggiungere ShardToZone	No
bilanciatore CollectionStatus	No
Balancer Start	No
Stato del bilanciatore	No
Balancer Stop	No

Comando	Cluster elastico
controllare ShardingIndex	No
chiaro JumboFlag	No
pulire Orphane	No
pulire ReshardCollection	No
commettere ReshardCollection	No
abilita il partizionamento	Sì
sciacquare RouterConfig	No
ottenere ShardMap	No
ottenere ShardVersion	No
isdbgrid	No
Elenca frammenti	No
Chiave mediana	No
Sposta Chunk	No
Sposta primario	No
Unisci blocchi	No
raffinare CollectionShardKey	No
Rimuovi Hard	No
rimuovere ShardFromZone	No
Collezione Reshard	No
set AllowMigrations	No

Comando	Cluster elastico
set ShardVersion	No
Collezione SHARD	Sì
Stato di condivisione	No
dividi	No
Vettore diviso	No
Deseleziona Sharding	No
aggiornamento ZoneKeyRange	No

Operatori di interrogazione e proiezione

Argomenti

- Operatori di array
- Operatori bit per bit
- Operatore di commento
- Operatori di confronto
- Operatori di elementi
- Operatori di interrogazione di valutazione
- Operatori logici
- Operatori di proiezione

Operatori di array

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$all	Sì	Sì	Sì	Sì	Sì
$elemMatch	Sì	Sì	Sì	Sì	Sì

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$size	Sì	Sì	Sì	Sì	Sì

Operatori bit per bit

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$ bit AllSet	Sì	Sì	Sì	Sì	Sì
$ bit AnySet	Sì	Sì	Sì	Sì	Sì
$ bit AllClear	Sì	Sì	Sì	Sì	Sì
$ bit AnyClear	Sì	Sì	Sì	Sì	Sì

Operatore di commento

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$comment	Sì	Sì	Sì	Sì	Sì

Operatori di confronto

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$eq	Sì	Sì	Sì	Sì	Sì
$gt	Sì	Sì	Sì	Sì	Sì
$gte	Sì	Sì	Sì	Sì	Sì

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$in	Sì	Sì	Sì	Sì	Sì
$lt	Sì	Sì	Sì	Sì	Sì
$lte	Sì	Sì	Sì	Sì	Sì
$ne	Sì	Sì	Sì	Sì	Sì
$nin	Sì	Sì	Sì	Sì	Sì

Operatori di elementi

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$exists	Sì	Sì	Sì	Sì	Sì
$type	Sì	Sì	Sì	Sì	Sì

Operatori di interrogazione di valutazione

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$expr	No	Sì	Sì	Sì	No
$jsonSchema	No	Sì	Sì	Sì	No
$mod	Sì	Sì	Sì	Sì	Sì
$regex	Sì	Sì	Sì	Sì	Sì
$text	No	No	Sì	Sì	No
$where	No	No	No	No	No

Operatori logici

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$and	Sì	Sì	Sì	Sì	Sì
$nor	Sì	Sì	Sì	Sì	Sì
$not	Sì	Sì	Sì	Sì	Sì
$or	Sì	Sì	Sì	Sì	Sì

Operatori di proiezione

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$	Sì	Sì	Sì	Sì	Sì
$elemMatch	Sì	Sì	Sì	Sì	Sì
$meta	No	No	Sì	Sì	No
$slice	Sì	Sì	Sì	Sì	Sì

Operatori di aggiornamento

Argomenti

- Operatori di array
- Operatori bit per bit
- Operatori di campo
- Modificatori di aggiornamento

Operatori di array

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$	Sì	Sì	Sì	Sì	Sì
$[]	Sì	Sì	Sì	Sì	Sì
$ [<identificatore>]	Sì	Sì	Sì	Sì	Sì
$addToSet	Sì	Sì	Sì	Sì	Sì
$pop	Sì	Sì	Sì	Sì	Sì
$pullAll	Sì	Sì	Sì	Sì	Sì
$pull	Sì	Sì	Sì	Sì	Sì
$push	Sì	Sì	Sì	Sì	Sì

Operatori bit per bit

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$bit	Sì	Sì	Sì	Sì	Sì

Operatori di campo

Operatore	3.6	4.0	5.0	8.0	Cluster elastico
$currentDate	Sì	Sì	Sì	Sì	Sì
$inc	Sì	Sì	Sì	Sì	Sì

Operatore	3.6	4.0	5.0	8.0	Cluster elastico
$max	Sì	Sì	Sì	Sì	Sì
$min	Sì	Sì	Sì	Sì	Sì
$mul	Sì	Sì	Sì	Sì	Sì
$rename	Sì	Sì	Sì	Sì	Sì
$set	Sì	Sì	Sì	Sì	Sì
$setOnInsert	Sì	Sì	Sì	Sì	Sì
$unset	Sì	Sì	Sì	Sì	Sì

Modificatori di aggiornamento

Operatore	3.6	4.0	5.0	8.0	Cluster elastico
$each	Sì	Sì	Sì	Sì	Sì
$position	Sì	Sì	Sì	Sì	Sì
$slice	Sì	Sì	Sì	Sì	Sì
$sort	Sì	Sì	Sì	Sì	Sì

Dati geospaziali

Specificatori di geometria

Selettori di query	3.6	4.0	5.0	8.0	Cluster elastico
$box	No	No	No	No	No

Selettori di query	3.6	4.0	5.0	8.0	Cluster elastico
$center	No	No	No	No	No
$centerSphere	No	No	No	No	No
$geometry	Sì	Sì	Sì	Sì	Sì
$maxDistance	Sì	Sì	Sì	Sì	Sì
$minDistance	Sì	Sì	Sì	Sì	Sì
$nearSphere	Sì	Sì	Sì	Sì	Sì
$polygon	No	No	No	No	No
$uniqueDocs	No	No	No	No	No

Selettori di interrogazione

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$geoIntersects	Sì	Sì	Sì	Sì	Sì
$geoWithin	Sì	Sì	Sì	Sì	Sì
$near	Sì	Sì	Sì	Sì	Sì
$nearSphere	Sì	Sì	Sì	Sì	Sì
$polygon	No	No	No	No	No
$uniqueDocs	No	No	No	No	No

Metodi Cursor

Comando	3.6	4.0	5.0	8.0	Cluster elastico
cursor.batchSize()	Sì	Sì	Sì	Sì	Sì
cursor.close()	Sì	Sì	Sì	Sì	Sì
cursor.collation()	No	No	No	Sì	No
cursor.comment()	Sì	Sì	Sì	Sì	Sì
cursor.count()	Sì	Sì	Sì	Sì	Sì
cursor.explain()	Sì	Sì	Sì	Sì	No
cursor.forEach()	Sì	Sì	Sì	Sì	Sì
cursor.hasNext()	Sì	Sì	Sì	Sì	Sì
cursor.hint()	Sì	Sì	Sì	Sì	Sì*
cursor.isClosed()	Sì	Sì	Sì	Sì	Sì
cursor.isExhausted()	Sì	Sì	Sì	Sì	No
cursor.itcount()	Sì	Sì	Sì	Sì	No
cursor.limit()	Sì	Sì	Sì	Sì	No

Comando	3.6	4.0	5.0	8.0	Cluster elastico
cursor.map()	Sì	Sì	Sì	Sì	No
cursor.max()	No	No	Sì (5.0.2+)	Sì (8.0.1+)	No
cursor.maxScan()	Sì	Sì	Sì	Sì	No
cursor.maxTimeMS()	Sì	Sì	Sì	Sì	No
cursor.min()	No	No	Sì (5.0.2+)	Sì (8.0.1+)	No
cursor.next()	Sì	Sì	Sì	Sì	Sì
cursore.no() CursorTimeout	No	No	No	No	No
cursore.objs() LeftInBatch	Sì	Sì	Sì	Sì	No
cursor.pretty()	Sì	Sì	Sì	Sì	No
cursor.readConcern()	Sì	Sì	Sì	Sì	No
cursor.readPref()	Sì	Sì	Sì	Sì	No
cursor.returnKey()	No	No	No	No	No
cursore.showRecordId ()	No	No	No	No	No
cursor.size()	Sì	Sì	Sì	Sì	No

Comando	3.6	4.0	5.0	8.0	Cluster elastico
cursor.skip()	Sì	Sì	Sì	Sì	No
cursor.sort()	Sì	Sì	Sì	Sì	No
cursor.tailable()	No	No	No	No	No
cursor.toArray()	Sì	Sì	Sì	Sì	No

* L'indice `hint` è supportato con le espressioni di indice. Ad esempio, `db.foo.find().hint({x:1})`.

I `cursor.max()` metodi `cursor.min()` and richiedono un nome `hint` che nomini l'indice da utilizzare. Entrambi i metodi devono specificare gli stessi campi e non è possibile combinarli con un `$natural` ordinamento. Sono supportati solo nel `find` comando.

Operatori di pipeline di aggregazione

Argomenti

- Espressioni accumulatrici
- Operatori aritmetici
- Operatori di array
- Operatori booleani
- Operatori di confronto
- Operatori di espressione condizionale
- Operatore del tipo di dati
- Operatore di dimensione dei dati
- Operatori di data
- Operatore letterale
- Operatore di fusione
- Operatore naturale

- Operatori su set
- Operatori di fase
- Operatori di stringa
- Variabili di sistema
- Operatore di ricerca testuale
- Operatori di conversione dei tipi
- Operatori variabili
- Operatori trigonometrici
- Operatori bit per bit (aggregazione)
- Operatori di timestamp
- Operatori vari

Espressioni accumulatrici

Expression	3.6	4.0	5.0	8.0	Cluster elastico
$accumulatore	-	-	No	No	No
$aggiungiToSet	Sì	Sì	Sì	Sì	Sì
$avg	Sì	Sì	Sì	Sì	Sì
$fondo	-	-	-	Sì (8.0.1+)	No
$ in basso N	-	-	-	Sì (8.0.1+)	No
$count	-	-	No	Sì (8.0.1+)	No
$covarianzaPOP	No	No	No	Sì (8.0.2+)	No
$covarianzeAMP	No	No	No	Sì (8.0.2+)	No

Expression	3.6	4.0	5.0	8.0	Cluster elastico
$DenseRank	No	No	No	Sì (8.0.2+)	No
$derivato	No	No	No	No	No
$Numero del documento	No	No	No	Sì (8.0.2+)	No
$exp MovingAvg	No	No	No	No	No
$first	Sì	Sì	Sì	Sì	Sì
$primoN	-	-	-	Sì (8.0.1+)	No
$integrale	No	No	No	No	No
$last	Sì	Sì	Sì	Sì	Sì
$ l'ultimo N	-	-	-	Sì (8.0.1+)	No
$LinearFill	No	No	No	No	No
$locf	No	No	No	No	No
$max	Sì	Sì	Sì	Sì	Sì
$max N	-	-	-	Sì (8.0.1+)	No
$mediana	-	-	-	Sì (8.0.1+)	No
$min	Sì	Sì	Sì	Sì	Sì
$minN	-	-	-	Sì (8.0.1+)	No
$percentile	-	-	-	Sì (8.0.1+)	No
$push	Sì	Sì	Sì	Sì	Sì
$rango	No	No	No	Sì (8.0.2+)	No

Expression	3.6	4.0	5.0	8.0	Cluster elastico
$turno	No	No	No	No	No
$standard DevPop	No	No	No	Sì (8.0.1+)	No
$standard DevSamp	No	No	No	Sì (8.0.1+)	No
$sum	Sì	Sì	Sì	Sì	Sì
$top	-	-	-	Sì (8.0.1+)	No
$ a PN	-	-	-	Sì (8.0.1+)	No

Operatori aritmetici

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$abs	Sì	Sì	Sì	Sì	Sì
$add	Sì	Sì	Sì	Sì	Sì
$ceil	No	Sì	Sì	Sì	Sì
$divide	Sì	Sì	Sì	Sì	Sì
$exp	No	Sì	Sì	Sì	Sì
$floor	No	Sì	Sì	Sì	Sì
$ln	No	Sì	Sì	Sì	Sì
$log	No	Sì	Sì	Sì	Sì
$log10	No	Sì	Sì	Sì	Sì

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$mod	Sì	Sì	Sì	Sì	Sì
$multiply	Sì	Sì	Sì	Sì	Sì
$pow	No	No	Sì (5.0.1+)	Sì	No
$round	-	-	No	Sì (8.0.1+)	No
$sqrt	No	Sì	Sì	Sì	Sì
$subtract	Sì	Sì	Sì	Sì	Sì
$trunc	No	No	No	Sì (8.0.1+)	No

Operatori di array

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$matrice ElemAt	Sì	Sì	Sì	Sì	Sì
$matrice ToObject	Sì	Sì	Sì	Sì	Sì
$concatAr rays	Sì	Sì	Sì	Sì	Sì
$filter	Sì	Sì	Sì	Sì	Sì
$first	-	-	Sì	Sì	No
$in	Sì	Sì	Sì	Sì	Sì
$indice OfArray	Sì	Sì	Sì	Sì	Sì

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$isArray	Sì	Sì	Sì	Sì	Sì
$last	-	-	Sì	Sì	No
$oggetto ToArray	Sì	Sì	Sì	Sì	Sì
$range	Sì	Sì	Sì	Sì	Sì
$reverseA rray	Sì	Sì	Sì	Sì	Sì
$sortArray	No	No	No	Sì	No
$reduce	Sì	Sì	Sì	Sì	Sì
$size	Sì	Sì	Sì	Sì	Sì
$slice	Sì	Sì	Sì	Sì	Sì
$zip	Sì	Sì	Sì	Sì	Sì

Operatori booleani

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$and	Sì	Sì	Sì	Sì	Sì
$not	Sì	Sì	Sì	Sì	Sì
$or	Sì	Sì	Sì	Sì	Sì

Operatori di confronto

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$cmp	Sì	Sì	Sì	Sì	Sì
$eq	Sì	Sì	Sì	Sì	Sì
$gt	Sì	Sì	Sì	Sì	Sì
$gte	Sì	Sì	Sì	Sì	Sì
$lt	Sì	Sì	Sì	Sì	Sì
$lte	Sì	Sì	Sì	Sì	Sì
$ne	Sì	Sì	Sì	Sì	Sì

Operatori di espressione condizionale

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$cond	Sì	Sì	Sì	Sì	Sì
$ifNull	Sì	Sì	Sì	Sì	Sì
$switch	No	Sì	Sì	Sì	No

Operatore del tipo di dati

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$type	Sì	Sì	Sì	Sì	Sì

Operatore di dimensione dei dati

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$Dimensione binaria	-	-	No	Sì (8.0.1+)	No
$bsonSize	-	-	No	Sì (8.0.1+)	No

Operatori di data

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$dataAggiungi	No	No	Sì	Sì	Sì
$dateDiff	-	-	Sì	Sì	No
$data FromParts	No	No	Sì (5.0.1+)	No	No
$data FromString	Sì	Sì	Sì	Sì	Sì
$dateSottrai	No	No	Sì	Sì	Sì
$data ToParts	No	No	Sì (5.0.1+)	No	No
$data ToString	Sì	Sì	Sì	Sì	Sì
$dateTrunc	-	-	No	Sì	No
$giorno OfMonth	Sì	Sì	Sì	Sì	Sì

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$giorno OfWeek	Sì	Sì	Sì	Sì	Sì
$giorno OfYear	Sì	Sì	Sì	Sì	Sì
$hour	Sì	Sì	Sì	Sì	Sì
$ iso DayOfWeek	Sì	Sì	Sì	Sì	Sì
$isoWeek	Sì	Sì	Sì	Sì	Sì
$ iso WeekYear	Sì	Sì	Sì	Sì	Sì
$millisecond	Sì	Sì	Sì	Sì	Sì
$minute	Sì	Sì	Sì	Sì	Sì
$month	Sì	Sì	Sì	Sì	Sì
$second	Sì	Sì	Sì	Sì	Sì
$week	Sì	Sì	Sì	Sì	Sì
$year	Sì	Sì	Sì	Sì	Sì

Operatore letterale

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$literal	Sì	Sì	Sì	Sì	Sì

Operatore di fusione

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$mergeObjects	Sì	Sì	Sì	Sì	Sì

Operatore naturale

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$naturale	Sì	Sì	Sì	Sì	Sì

Operatori su set

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$ tutti ElementsTrue	No	Sì	Sì	Sì	Sì
$qualsiasi ElementTrue	No	Sì	Sì	Sì	Sì
$setDifference	No	Sì	Sì	Sì	Sì
$setEquals	Sì	Sì	Sì	Sì	Sì
$setIntersection	Sì	Sì	Sì	Sì	Sì
$set IsSubset	Sì	Sì	Sì	Sì	Sì

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$setUnion	Sì	Sì	Sì	Sì	Sì

Operatori di fase

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$addFields	Sì	Sì	Sì	Sì	Sì
$bucket	No	No	No	Sì	No
$bucketAuto	No	No	No	Sì (8.0.2+)	No
$Cambia stream	Sì	Sì	Sì	Sì	No
$modifica StreamSplitLargeEvent	No	No	No	Sì (8.0.2+)	No
$collStats	No	Sì	Sì	Sì	No
$count	Sì	Sì	Sì	Sì	Sì
$currentOp	Sì	Sì	Sì	Sì	Sì
$ densificare	No	No	No	No	No
$facet	No	No	No	Sì (8.0.2+)	No
$compila	No	No	No	No	No
$geoNear	Sì	Sì	Sì	Sì	Sì
$graphLookup	No	No	No	Sì	No

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$group	Sì	Sì	Sì	Sì	Sì
$indexStats	Sì	Sì	Sì	Sì	No
$limit	Sì	Sì	Sì	Sì	Sì
$lista LocalSessions	No	No	No	No	No
$listSessions	No	No	No	No	No
$lookup	Sì	Sì	Sì	Sì	Sì
$match	Sì	Sì	Sì	Sì	Sì
$unisci	-	-	No	Sì	No
$out	Sì	Sì	Sì	Sì	No
$piano CacheStats	-	-	No	No	No
$project	Sì	Sì	Sì	Sì	Sì
$redact	Sì	Sì	Sì	Sì	Sì
$replaceRoot	Sì	Sì	Sì	Sì	Sì
$sample	Sì	Sì	Sì	Sì	Sì
$set	-	-	No	Sì	No
$set WindowFields	-	-	No	Sì (8.0.2+)	No
$skip	Sì	Sì	Sì	Sì	Sì

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$sort	Sì	Sì	Sì	Sì	Sì
$ordinaByCount	No	No	No	Sì (8.0.1+)	No
$Union con	-	-	No	No	No
$unset	-	-	No	Sì	No
$unwind	Sì	Sì	Sì	Sì	Sì
$sostituisci con	No	No	No	Sì	No
$ Ricerca vettoriale	No	No	No	Sì	No
elenco $ SearchIndexes	-	-	-	Sì (8.0.1+)	No

Operatori di stringa

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$concat	Sì	Sì	Sì	Sì	Sì
$indiceOfBytes	Sì	Sì	Sì	Sì	Sì
$indexOfCP	Sì	Sì	Sì	Sì	Sì
$ltrim	No	Sì	Sì	Sì	No
$regexFind	-	-	Sì	Sì	No

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$ espressione regolare FindAll	-	-	Sì	Sì	No
$RegexMatch	-	-	Sì	Sì	No
$ Sostituisci tutto	-	-	Sì	Sì	No
$ Sostituisci uno	-	-	Sì	Sì	No
$rtrim	No	Sì	Sì	Sì	No
$split	Sì	Sì	Sì	Sì	Sì
$strcasecmp	Sì	Sì	Sì	Sì	Sì
$star LenBytes	Sì	Sì	Sì	Sì	Sì
$strLenCP	Sì	Sì	Sì	Sì	Sì
$substr	Sì	Sì	Sì	Sì	Sì
$substrBytes	Sì	Sì	Sì	Sì	Sì
$substrCP	Sì	Sì	Sì	Sì	Sì
$toLower	Sì	Sì	Sì	Sì	Sì
$toUpper	Sì	Sì	Sì	Sì	Sì
$trim	No	Sì	Sì	Sì	No

Variabili di sistema

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$$CURRENT	No	No	No	No	No
$$DESCEND	Sì	Sì	Sì	Sì	Sì
$$KEEP	Sì	Sì	Sì	Sì	Sì
$$PRUNE	Sì	Sì	Sì	Sì	Sì
$$REMOVE	No	No	No	No	No
$RADICE	Sì	Sì	Sì	Sì	Sì

Operatore di ricerca testuale

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$meta	No	No	Sì	Sì	No
$ricerca	No	No	Sì	Sì	No

Operatori di conversione dei tipi

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$converti	No	Sì	Sì	Sì	Sì
$è il numero	-	-	No	Sì (8.0.1+)	No
$ a BOOL	No	Sì	Sì	Sì	Sì

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$ fino ad oggi	No	Sì	Sì	Sì	Sì
$ a Decimal	No	Sì	Sì	Sì	Sì
$ a Double	No	Sì	Sì	Sì	Sì
$ a INT	No	Sì	Sì	Sì	Sì
$ a Long	No	Sì	Sì	Sì	Sì
$a ObjectId	No	Sì	Sì	Sì	Sì
$toString	No	Sì	Sì	Sì	Sì
da $ a UUID	-	-	-	Sì (8.0.1+)	No

Operatori variabili

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$let	Sì	Sì	Sì	Sì	Sì
$map	Sì	Sì	Sì	Sì	Sì

Operatori trigonometrici

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$acos	-	-	-	Sì (8.0.1+)	No
$acosh	-	-	-	Sì (8.0.1+)	No
$asin	-	-	-	Sì (8.0.1+)	No

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$asinh	-	-	-	Sì (8.0.1+)	No
$atan	-	-	-	Sì (8.0.1+)	No
$atan2	-	-	-	Sì (8.0.1+)	No
$atanh	-	-	-	Sì (8.0.1+)	No
$cos	-	-	-	Sì (8.0.1+)	No
$cosh	-	-	-	Sì (8.0.1+)	No
$gradiToRadians	-	-	-	Sì (8.0.1+)	No
$radiantiToDegrees	-	-	-	Sì (8.0.1+)	No
$sin	-	-	-	Sì (8.0.1+)	No
$sinh	-	-	-	Sì (8.0.1+)	No
$tan	-	-	-	Sì (8.0.1+)	No
$tanh	-	-	-	Sì (8.0.1+)	No

Operatori bit per bit (aggregazione)

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$BIT e	-	-	-	Sì (8.0.1+)	No
$bitNot	-	-	-	Sì (8.0.1+)	No
$BIT o	-	-	-	Sì (8.0.1+)	No

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$BitXor	-	-	-	Sì (8.0.1+)	No

Operatori di timestamp

Comando	3.6	4.0	5.0	8.0	Cluster elastico
Incremento $TS	-	-	-	Sì (8.0.1+)	No
$tsSecondo	-	-	-	Sì (8.0.1+)	No

Operatori vari

Comando	3.6	4.0	5.0	8.0	Cluster elastico
$getField	-	-	Sì	Sì	No
$ rand	-	-	Sì (5.0.1+)	Sì	No
$Frequenza di campionamento	-	-	No	Sì (8.0.1+)	No
$sigmoide	-	-	-	Sì (8.0.1+)	No

Tipi di dati

Comando	3.6	4.0	5.0	8.0	Cluster elastico
Numero intero a 32 bit (int)	Sì	Sì	Sì	Sì	Sì
Numero intero a 64 bit (lungo)	Sì	Sì	Sì	Sì	Sì
Array	Sì	Sì	Sì	Sì	Sì
Dati binari	Sì	Sì	Sì	Sì	Sì
Booleano	Sì	Sì	Sì	Sì	Sì
Data	Sì	Sì	Sì	Sì	Sì
DBPointer	No	No	No	No	No
DBRefs	No	No	No	No	No
Decimal128	Sì	Sì	Sì	Sì	Sì
Double	Sì	Sì	Sì	Sì	Sì
JavaScript	No	No	No	No	No
JavaScript(con ambito)	No	No	No	No	No
MaxKey	Sì	Sì	Sì	Sì	Sì
MinKey	Sì	Sì	Sì	Sì	Sì
Null	Sì	Sì	Sì	Sì	Sì
Oggetto	Sì	Sì	Sì	Sì	Sì

Comando	3.6	4.0	5.0	8.0	Cluster elastico
ObjectId	Sì	Sì	Sì	Sì	Sì
Espressione regolare	Sì	Sì	Sì	Sì	Sì
Stringa	Sì	Sì	Sì	Sì	Sì
Symbol	No	No	No	No	No
Time stamp	Sì	Sì	Sì	Sì	Sì
Undefined	No	No	No	No	No

Indici e proprietà degli indici

Argomenti

- Indici
- Proprietà dell'indice

Indici

Comando	3.6	4.0	5.0	8.0	Cluster elastico
2dsphere	Sì	Sì	Sì	Sì	Sì
Indice 2d	No	No	No	No	No
Indice composto	Sì	Sì	Sì	Sì	Sì
Indice con hash	No	No	No	No	No

Comando	3.6	4.0	5.0	8.0	Cluster elastico
Indice con più chiavi	Sì	Sì	Sì	Sì	Sì
Indice con campo singolo	Sì	Sì	Sì	Sì	Sì
Indice di testo	No	No	Sì	Sì	No
Carattere jolly	No	No	No	No	No

Proprietà dell'indice

Comando	3.6	4.0	5.0	8.0	Cluster elastico
Contesto	Sì	Sì	Sì	Sì	Sì
Senza distinzione tra maiuscole e minuscole	No	No	No	Sì	No
Hidden	No	No	No	Sì (8.0.1+)	No
Parziale	No	No	Sì	Sì	No
Sparse	Sì	Sì	Sì	Sì	Sì
Testo	No	No	Sì	Sì	No
TTL	Sì	Sì	Sì	Sì	Sì
Unique	Sì	Sì	Sì	Sì	Sì
Vettore	No	No	Sì	Sì	No

Per informazioni dettagliate su operatori MongoDB specifici, consultate i seguenti argomenti:

- Operatori di pipeline di aggregazione
- Dati geospaziali
- Operatori di proiezione
- Aggiorna operatori

Operatori di pipeline di aggregazione

Questa sezione fornisce informazioni dettagliate sugli operatori di pipeline di aggregazione supportati da Amazon DocumentDB.

Argomenti

- $$DESCEND
- $$KEEP
- $$PRUNE
- $RADICE
- $abs
- $acos
- $acosh
- $add
- $aggiungi ToSet
- $addFields
- $ tutti ElementsTrue
- $and
- $qualsiasi ElementTrue
- $matrice ElemAt
- $matrice ToObject
- $asin
- $asinh
- $atan
- $atan2

- $atanh
- $avg
- $Dimensione binaria
- $BIT e
- $bitNot
- $BIT o
- $BitXor
- $fondo
- $ in basso N
- $bsonSize
- $bucket
- $bucketAuto
- $ceil
- $Cambia stream
- $modifica StreamSplitLargeEvent
- $cmp
- $collStats
- $concat
- $concatArrays
- $cond
- $converti
- $cos
- $cosh
- $count (fase)
- $count (accumulatore)
- $covarianza POP
- $covarianzeAMP
- $currentOp
- $dataAggiungi

- $dateDiff
- $data FromParts
- $data FromString
- $dateSottrai
- $data ToParts
- $data ToString
- $dateTrunc
- $giorno OfMonth
- $giorno OfWeek
- $giorno OfYear
- $gradi ToRadians
- $DenseRank
- $divide
- $Numero del documento
- $eq
- $exp
- $facet
- $filter
- $first
- $primoN
- $floor
- $geoNear
- $graphLookup
- $group
- $gt
- $gte
- $hour
- $ifNull
- $in

- $indice OfArray
- $indice OfBytes
- $indexOfCP
- $indexStats
- $isArray
- $è il numero
- $ iso DayOfWeek
- $isoWeek
- $ iso WeekYear
- $last
- $ l'ultimo N
- $let
- $limit
- elenco $ SearchIndexes
- $literal
- $ln
- $log
- $log10
- $lt
- $lte
- $lookup
- $ltrim
- $map
- $match
- $max
- $max N
- $mediana
- $meta
- $unisci

- $mergeObjects
- $millisecond
- $min
- $minN
- $minute
- $mod
- $month
- $multiply
- $naturale
- $ne
- $not
- $oggetto ToArray
- $or
- $out
- $percentile
- $pow
- $push
- $project
- $radianti ToDegrees
- $ rand
- $range
- $rango
- $redact
- $reduce
- $regexFind
- $ espressione regolare FindAll
- $RegexMatch
- $ Sostituisci tutto
- $ Sostituisci uno

- $replaceRoot
- $sostituisci con
- $reverseArray
- $round
- $rtrim
- $sample
- $ricerca
- $second
- $set
- $setDifference
- $setEquals
- $setIntersection
- $set IsSubset
- $setUnion
- $set WindowFields
- $sigmoide
- $sin
- $sinh
- $skip
- $slice
- $size
- $sort
- $sortArray
- $ordina ByCount
- $split
- $sqrt
- $standard DevPop
- $standard DevSamp
- $star LenBytes

- $strLenCP
- $strcasecmp
- $substr
- $substrBytes
- $substrCP
- $subtract
- $sum
- $switch
- $tan
- $tanh
- $top
- $ a PN
- $trunc
- Incremento $TS
- $tsSecondo
- $ a BOOL
- $ fino ad oggi
- $ a Decimal
- $ a Double
- $ a INT
- $ a Long
- $toLower
- $a ObjectId
- $toString
- $toUpper
- da $ a UUID
- $trim
- $type
- $unset
- $unwind

- $ Ricerca vettoriale
- $week
- $year
- $zip

$$DESCEND

L'$$DESCENDoperatore in Amazon DocumentDB è uno speciale operatore di array posizionale utilizzato nella `$redact` fase di pipeline. Indica alla pipeline di aggregazione di scendere nel documento corrente ed elaborare tutti i campi, indipendentemente dal loro livello di nidificazione.

Quando lo `$redact` stage incontra l'$$DESCENDoperatore, manterrà visibili tutti i campi del documento corrente e li elaborerà ulteriormente lungo la pipeline. Ciò è utile quando si desidera oscurare o eliminare selettivamente determinati campi in base a una condizione, mantenendo al contempo la struttura del documento.

Parametri

Nessuna.

Esempio (MongoDB Shell)

In questo esempio, useremo lo `$redact` stage con l'$$DESCENDoperatore per visualizzare selettivamente i documenti in cui il `code` campo è uguale a «Reg».

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.patient.aggregate([
```

```
  { $redact: {
    $cond: {
      if: { $eq: ["Reg", "$code"] },
      then: "$$DESCEND",
      else: "$$PRUNE"
    }
  }}
]);
```

Output

```
{
  "_id": 2,
  "code": "Reg",
  "patient": "Jane Doe",
  "DOB": "3/27/1989",
  "Hospital": "Cherry Hill"
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$$DESCEND` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

La variabile di `$$KEEP` sistema viene utilizzata con lo `$redact` stage nella pipeline di aggregazione per mantenere invariato il documento o il campo corrente e includerlo nell'output.

Parametri

Nessuno

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo di $$KEEP in una pipeline di aggregazione Amazon DocumentDB. I documenti vengono conservati solo se l'accesso è uguale a «pubblico», altrimenti vengono rimossi.

Crea documenti di esempio

```
db.articles.insertMany([
  { title: "Article A", access: "public", content: "Visible content" },
  { title: "Article B", access: "private", content: "Hidden content" }
]);
```

Esempio di interrogazione

```
db.articles.aggregate([
  {
    $redact: {
      $cond: [
        { $eq: ["$access", "public"] },
        "$$KEEP",
        "$$PRUNE"
      ]
    }
  }
]);
```

Output

```
[
  {
    "_id" : ObjectId("..."),
    "title" : "Article A",
    "access" : "public",
    "content" : "Visible content"
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del $$KEEP comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

La variabile di $$PRUNE sistema viene utilizzata con la `$redact` fase della pipeline di aggregazione per escludere documenti o livelli di documento incorporati dai risultati. Quando una condizione restituisce un risultato positivo$$PRUNE, il documento o il documento secondario corrente viene rimosso dall'output. In genere viene utilizzato con $$DESCEND (per conservare e scorrere il documento) o $$KEEP (per conservare il documento a tutti i livelli).

Parametri

Nessuna. La variabile di $$PRUNE sistema viene utilizzata senza parametri e deve essere utilizzata con. `$redact`

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare $$PRUNE with per escludere gli utenti di età superiore `$redact` ai 30 anni dai risultati.

Crea documenti di esempio

```
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" } }
])
```

Esempio di interrogazione

```
db.users.aggregate([
  {
    $redact: {
      $cond: {
        if: { $gt: ["$age", 30] },
        then: "$$PRUNE",
        else: "$$DESCEND"
      }
    }
  }
])
```

Output

```
[
  {
    "_id": 2,
    "name": "Saanvi Sarkar",
    "age": 28,
    "address": {
      "street": "456 Oak Rd",
      "city": "Someplace",
      "state": "NY"
    }
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del $$PRUNE comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$RADICE

L'$R00Toperatore in Amazon DocumentDB viene utilizzato per fare riferimento all'intero documento di input all'interno di una pipeline di aggregazione. Consente di accedere e manipolare l'intero documento, inclusi tutti i campi e le strutture annidati.

Parametri

Nessuno

Esempio (MongoDB Shell)

Questo esempio dimostra l'utilizzo $R00T per creare un registro di controllo che acquisisca il documento originale completo insieme ai metadati relativi a quando è stato elaborato.

Crea documenti di esempio

```
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
  }
]);
```

Esempio di query

```
db.orders.aggregate([
  {
    $project: {
      processedAt: new Date(),
      originalDocument: "$$ROOT",
      summary: {
        $concat: [
          "Order ",
          "$orderId",
          " for ",
          "$customer",
          " - Total: $",
          { $toString: "$totalAmount" }
        ]
      }
    }
  }
]);
```

Output

```
[
  {
    _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'
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del $R00T comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$abs`operatore in Amazon DocumentDB restituisce il valore assoluto di un numero. Può essere utilizzato nella pipeline di aggregazione per eseguire operazioni matematiche su campi numerici.

Parametri

- `number`: L'espressione numerica per la quale verrà restituito il valore assoluto.

Esempio (MongoDB Shell)

Questo esempio dimostra l'utilizzo dell'`$abs`operatore per trovare il valore assoluto di un campo numerico.

Crea documenti di esempio

```
db.numbers.insertMany([
  { "_id": 1, "value": -5 },
  { "_id": 2, "value": 10 },
  { "_id": 3, "value": -3.14 },
  { "_id": 4, "value": 0 }
]);
```

Esempio di query

```
db.numbers.aggregate([
```

```
  { $project: {
    "_id": 1,
    "absolute_value": { $abs: "$value" }
  }}
]);
```

Output

```
[
  { "_id": 1, "absolute_value": 5 },
  { "_id": 2, "absolute_value": 10 },
  { "_id": 3, "absolute_value": 3.14 },
  { "_id": 4, "absolute_value": 0 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$abs` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

Nuovo dalla versione 8.0.1.

L'`$acos`operatore in Amazon DocumentDB restituisce l'arcocoseno (coseno inverso) di un valore. Il valore di input deve essere compreso tra -1 e 1.

Parametri

- `expression`: Un'espressione che si risolve in un numero compreso tra -1 e 1.

Il risultato è in radianti. Per ottenere gradi, applicali `$radiansToDegrees` all'output.

Il tipo restituito è `double` predefinito. Se l'input è un decimale a 128 bit, anche l'output è un decimale a 128 bit.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$acos`operatore per calcolare l'arcocoseno di valori numerici.

Crea documenti di esempio

```
db.values.insertMany([
  { "_id": 1, "value": 1 },
  { "_id": 2, "value": 0 },
  { "_id": 3, "value": -1 }
]);
```

Esempio di interrogazione

```
db.values.aggregate([
  { $project: {
    "angle": { $acos: "$value" }
  }}
]);
```

Output

```
[
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 1.5707963267948966 },
  { "_id": 3, "angle": 3.141592653589793 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$acos` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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()
```

$acosh

Nuovo dalla versione 8.0.1.

L'`$acosh`operatore in Amazon DocumentDB restituisce il coseno iperbolico inverso (arccoseno iperbolico) di un valore. Il valore di input deve essere maggiore o uguale a 1.

Parametri

- `expression`: Un'espressione che si risolve in un numero maggiore o uguale a 1.

Il tipo restituito è predefinito`double`. Se l'input è un decimale a 128 bit, anche l'output è un decimale a 128 bit.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$acosh`operatore per calcolare il coseno iperbolico inverso di valori numerici.

Crea documenti di esempio

```
db.values.insertMany([
  { "_id": 1, "value": 1 },
  { "_id": 2, "value": 2 },
  { "_id": 3, "value": 10 }
]);
```

Esempio di interrogazione

```
db.values.aggregate([
  { $project: {
    "result": { $acosh: "$value" }
  }}
]);
```

Output

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 1.3169578969248166 },
  { "_id": 3, "result": 2.993222846126381 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$acosh` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

L'`$add`operatore in Amazon DocumentDB viene utilizzato per sommare numeri o date. Può essere utilizzato per eseguire operazioni aritmetiche su campi numerici o per eseguire operazioni aritmetiche di date aggiungendo un numero di unità di tempo a un campo di data.

Parametri

- `expression1`: Il primo numero o data da aggiungere.
- `expression2`: Il secondo numero o data da aggiungere.
- `expression3`: (opzionale) Numeri o date aggiuntivi da aggiungere.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$add`operatore per sommare due numeri.

Crea documenti di esempio

```
db.numbers.insertMany([
  { _id: 1, a: 5, b: 3 },
  { _id: 2, a: 10, b: 7 },
  { _id: 3, a: 2, b: 8 }
]);
```

Esempio di interrogazione

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    sum: { $add: ["$a", "$b"] }
  }}
])
```

Output

```
[
  { "_id" : 1, "sum" : 8 },
  { "_id" : 2, "sum" : 17 },
  { "_id" : 3, "sum" : 10 }
]
```

In questo esempio, l'`$add`operatore viene utilizzato per aggiungere i valori dei b campi a e per ogni documento e memorizzare il risultato nel `sum` campo.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$add` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$aggiungi ToSet

L'operatore `$addToSet` di aggregazione restituisce una matrice di valori univoci da un'espressione specificata per ogni gruppo. Viene utilizzato all'interno dello `$group` stage per accumulare valori distinti, eliminando automaticamente i duplicati.

Parametri

- `expression`: L'espressione da valutare per ogni documento del gruppo.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$addToSet`operatore per raccogliere città uniche in cui sono stati effettuati gli ordini per ciascun cliente.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.orders.aggregate([
  {
    $group: {
      _id: "$customer",
      cities: { $addToSet: "$city" }
    }
  }
]);
```

Output

```
[
  { _id: 'Bob', cities: [ 'Seattle', 'Boston' ] },
  { _id: 'Alice', cities: [ 'Seattle', 'Portland' ] }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$addToSet`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce una matrice dei valori univoci dell'espressione specificata per i documenti in ogni finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

> **Note**
>
> Se utilizzato come operatore di finestra in`$setWindowFields`, `$addToSet` è limitato a 100 MB di dati intermedi. Un'operazione che supera questo limite restituisce un errore.

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`customer`, ordina ogni partizione in base `day` e restituisce il set di `city` valori univoci dall'inizio della partizione fino al documento corrente.

```
db.visits.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$customer",
      sortBy: { day: 1 },
      output: {
        citiesSoFar: {
          $addToSet: "$city",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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" ] }
]
```

Ogni documento è arricchito con `citiesSoFar` l'insieme di `city` valori univoci all'interno della sua partizione dall'inizio fino al documento corrente. Come nel modulo accumulatore, l'ordine dei valori nell'array restituito non è garantito.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$addToSet` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano sia l'utilizzo dell'accumulatore (in`$group`) che l'utilizzo dell'operatore di finestra (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

La `$addFields` fase della pipeline di aggregazione di Amazon DocumentDB consente di aggiungere nuovi campi calcolati ai documenti. Questo può essere utile per aggiungere dati derivati o trasformati ai documenti.

Parametri

- `newField`: nome del nuovo campo da aggiungere.
- `expression`: espressione che si risolve nel valore del nuovo campo.

Esempio (MongoDB Shell)

L'esempio seguente mostra come `$addFields` aggiungere un nuovo campo `TotalInventory` che calcola l'inventario totale in base ai campi and. `Inventory.OnHand` `Inventory.OrderQnty`

Crea documenti di esempio

```
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
  }
]);
```

Esempio di interrogazione

```
db.example.aggregate([
  {
    $addFields: {
      TotalInventory: { $add: ["$Inventory.OnHand", "$Inventory.OrderQnty"] }
    }
  }
```

```
])
```

Output

```
[
  {
    "_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
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$addFields` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$ tutti ElementsTrue

Nuovo dalla versione 4.0

L'`$allElementsTrue`operatore viene utilizzato per verificare se tutti gli elementi di un array restituiscono un valore vero.

Parametri

- `expression`: espressione che restituisce un array.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo di `$allElementsTrue` per verificare se tutti gli elementi di un array sono veri.

Crea documenti di esempio

```
db.collection.insert([
  { "name": "John", "scores": [100, 90, 80] },
  { "name": "Jane", "scores": [80, 85, 0] },
  { "name": "Bob", "scores": [90, 95, null] }
])
```

Esempio di interrogazione

```
db.collection.find({
  "scores": { "$allElementsTrue": [{ "$gt": 0 }] }
})
```

Output

```
[
  { "_id" : ObjectId("..."), "name" : "John", "scores" : [ 100, 90, 80 ] },
  { "_id" : ObjectId("..."), "name" : "Bob", "scores" : [ 90, 95, null ] }
]
```

In questo esempio, la query verifica se tutti gli elementi dell'`scores`array sono maggiori di 0. Il documento con `"name": "Jane"` è escluso perché l'`scores`array contiene uno 0, che è un valore falso.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$allElementsTrue` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'operatore di `$and` aggregazione valuta più espressioni e restituisce `true` solo se tutte le espressioni restituiscono a. `true` Se un'espressione lo è`false`, restituisce. `false`

Parametri

- `expressions`: Una serie di espressioni da valutare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo `$and` dell'operatore per verificare se i prodotti soddisfano più criteri.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.products.aggregate([
  {
    $project: {
      name: 1,
      price: 1,
      inStock: 1,
      affordable: {
        $and: [
          { $lt: ["$price", 100] },
          { $eq: ["$inStock", true] }
        ]
      }
    }
  }
]);
```

Output

```
[
  { _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 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di $and aggregazione, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

$qualsiasi ElementTrue

Nuovo dalla versione 4.0

L'`$anyElementTrue`operatore viene utilizzato per determinare se un elemento di un array è vero.

Parametri

- `field`: un campo di matrice da valutare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo di `$anyElementTrue` per verificare se un elemento in un array è vero.

Crea documenti di esempio

```
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] }
])
```

Esempio di interrogazione

```
db.grades.aggregate([
  {
    $project: {
      student: 1,
      isAnyTrue: { $anyElementTrue: ["$scores"] },
      _id: 0
    }
  }
])
```

Output

```
[
  { student: 'Tim', isAnyTrue: false },
  { student: 'Bob', isAnyTrue: false },
  { student: 'Ivy', isAnyTrue: true }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$anyElementTrue` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$matrice ElemAt

L'`$arrayElemAt`operatore in Amazon DocumentDB consente di recuperare un elemento da un array in base alla sua posizione di indice. Ciò è particolarmente utile quando è necessario accedere a un elemento specifico all'interno di un campo di matrice nei documenti.

Parametri

- `array`: L'array di input da cui deve essere recuperato un elemento.
- `index`: la posizione dell'indice a base zero dell'elemento da recuperare. Questo valore deve essere un numero intero non negativo.

Esempio (MongoDB Shell)

In questo esempio, dimostreremo come utilizzare l'`$arrayElemAt`operatore per recuperare elementi specifici dall'`flight_miles`array della raccolta. `miles`

Crea documenti di esempio

```
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 ]}
]);
```

Esempio di interrogazione

```
db.miles.aggregate([
  { $project: {
    "_id": 1,
    "first_mile": { $arrayElemAt: [ "$flight_miles", 0 ] },
    "last_mile": { $arrayElemAt: [ "$flight_miles", -1 ] }
  }}
]);
```

Output

```
{ "_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 questo esempio, utilizziamo l'`$arrayElemAt`operatore per recuperare il primo e l'ultimo elemento dell'`flight_miles`array per ogni documento.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$arrayElemAt` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$matrice ToObject

L'`$arrayToObject`operatore in Amazon DocumentDB è l'opposto dell'`$objectToArray`operatore. Prende una serie di documenti di coppia chiave-valore e li converte in un unico documento. Ciò è particolarmente utile quando è necessario riconvertire una serie di coppie chiave-valore in una struttura di oggetto o documento.

Parametri

- `array expression`: espressione che si risolve in un array. Gli elementi dell'array devono essere documenti con due campi: k (la chiave) e v (il valore).

Esempio (MongoDB Shell)

L'esempio seguente mostra come `$arrayToObject` riconvertire una matrice di coppie chiave-valore in un documento.

Crea documenti di esempio

```
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 } }
]);
```

Esempio di interrogazione

```
db.videos.aggregate([
  { $project: { name: 1, videos: { $objectToArray: "$inventory" } } },
  { $project: { name: 1, inventory: { $arrayToObject: "$videos" } } }
]);
```

Output

```
{ "_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 questo esempio, l'`$objectToArray`operatore viene utilizzato per convertire l'`inventory`oggetto in una matrice di coppie chiave-valore. L'`$arrayToObject`operatore viene quindi utilizzato per riconvertire l'array in un documento, ripristinando la struttura originale dell'oggetto.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$arrayToObject` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

Nuovo dalla versione 8.0.1.

L'`$asin`operatore in Amazon DocumentDB restituisce l'arcoseno (seno inverso) di un valore. Il valore di input deve essere compreso tra -1 e 1.

Parametri

- `expression`: Un'espressione che si risolve in un numero compreso tra -1 e 1.

Il risultato è in radianti. Per ottenere gradi, applicali `$radiansToDegrees` all'output.

Il tipo restituito è `double` predefinito. Se l'input è un decimale a 128 bit, anche l'output è un decimale a 128 bit.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$asin`operatore per calcolare l'arcoseno di valori numerici.

Crea documenti di esempio

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 0.5 },
  { "_id": 3, "value": 1 }
]);
```

Esempio di interrogazione

```
db.values.aggregate([
  { $project: {
    "angle": { $asin: "$value" }
  }}
```

```
]);
```

Output

```
[
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.5235987755982989 },
  { "_id": 3, "angle": 1.5707963267948966 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$asin` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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()
```

$asinh

Nuovo dalla versione 8.0.1.

L'`$asinh`operatore in Amazon DocumentDB restituisce il seno iperbolico inverso di un valore. Utilizzalo nella pipeline di aggregazione per eseguire calcoli trigonometrici iperbolici su campi numerici.

Parametri

- `expression`: Un'espressione che si risolve in un numero.

Il risultato è in radianti. Per ottenere gradi, applicali `$radiansToDegrees` all'output.

Il tipo restituito è `double` predefinito. Se l'input è un decimale a 128 bit, anche l'output è un decimale a 128 bit.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$asinh`operatore per calcolare il seno iperbolico inverso dei valori numerici.

Crea documenti di esempio

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Esempio di interrogazione

```
db.values.aggregate([
  { $project: {
    "result": { $asinh: "$value" }
  }}
]);
```

Output

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 0.881373587019543 },
  { "_id": 3, "result": -0.881373587019543 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$asinh` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

Nuovo dalla versione 8.0.1.

L'`$atan`operatore in Amazon DocumentDB restituisce la tangente inversa (arcotangente) di un valore.

L'espressione di input deve essere convertita in un valore numerico.

Il risultato è in radianti. Per ottenere gradi, applicali `$radiansToDegrees` all'output.

Il tipo restituito è `double` predefinito. Se l'input è un decimale a 128 bit, anche l'output è un decimale a 128 bit.

Parametri

- `expression`: Un'espressione che si risolve in un numero.

Comportamento

null e NaN

Esempio	Risultati
`{ $atan: NaN }`	`NaN`
`{ $atan: null }`	`null`

Quando l'input è `null` o il campo di riferimento è mancante, il risultato è. `null` Un input di `NaN` produce`NaN`.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$atan`operatore per calcolare l'arcotangente di un valore.

Crea documenti di esempio

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Esempio di interrogazione

```
db.values.aggregate([
  { $project: {
    "angle": { $atan: "$value" }
  }}
```

```
]);
```

Output

```
[
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.7853981633974483 },
  { "_id": 3, "angle": -0.7853981633974483 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$atan` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

Nuovo dalla versione 8.0.1.

L'`$atan2`operatore in Amazon DocumentDB restituisce la tangente inversa (arcotangente) di`y / x`, dove `y` e `x` sono rispettivamente il primo e il secondo valore passati all'espressione.

Entrambe le espressioni devono essere convertite in valori numerici. L'operatore utilizza il segno di entrambi gli argomenti per determinare il quadrante corretto.

Il risultato è in radianti. Per ottenere gradi, applicali `$radiansToDegrees` all'output.

Il tipo restituito è `double` predefinito. Se gli input sono decimali a 128 bit, anche l'output è un decimale a 128 bit.

Parametri

- `expression 1`(y): Un'espressione che si risolve in un numero che rappresenta la coordinata y.
- `expression 2`(x): Un'espressione che si risolve in un numero che rappresenta la coordinata x.

Comportamento

null e NaN

Se uno degli argomenti è `null` (o fa riferimento a un campo mancante), il risultato è`null`. Se uno degli argomenti è`NaN`, il risultato è`NaN`. Quando un argomento è `null` e l'altro è`NaN`, ha la `null` precedenza.

Esempio	Risultati
`{ $atan2: [ NaN, <value> ] }`	`NaN`
`{ $atan2: [ <value>, NaN ] }`	`NaN`
`{ $atan2: [ null, <value> ] }`	`null`
`{ $atan2: [ <value>, null ] }`	`null`
`{ $atan2: [ NaN, null ] }`	`null`
`{ $atan2: [ null, NaN ] }`	`null`

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$atan2`operatore per calcolare l'arcotangente di y/x per un insieme di punti.

Crea documenti di esempio

```
db.points.insertMany([
  { "_id": 1, "y": 1, "x": 1 },
  { "_id": 2, "y": 1, "x": 0 },
  { "_id": 3, "y": -1, "x": -1 }
]);
```

Esempio di interrogazione

```
db.points.aggregate([
  { $project: {
    "angle": { $atan2: ["$y", "$x"] }
  }}
]);
```

Output

```
[
  { "_id": 1, "angle": 0.7853981633974483 },
  { "_id": 2, "angle": 1.5707963267948966 },
  { "_id": 3, "angle": -2.356194490192345 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$atan2` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

Nuovo dalla versione 8.0.1.

L'`$atanh`operatore in Amazon DocumentDB restituisce la tangente iperbolica inversa (arcotangente iperbolica) di un valore.

L'espressione di input deve essere costituita da un numero compreso tra 0 e (inclusi). `-1` `1` I valori al di fuori di questo intervallo causano un errore.

Il tipo restituito è `double` predefinito. Se l'input è un decimale a 128 bit, anche l'output è un decimale a 128 bit.

Parametri

- `expression`: Un'espressione che si risolve in un numero compreso tra e, incluso. `-1` `1`

Comportamento

null, NaN e +/- Infinity

Esempio	Risultati
`{ $atanh: NaN }`	`NaN`
`{ $atanh: null }`	`null`

Esempio	Risultati
`{ $atanh: 1 }`	`Infinity`
`{ $atanh: -1 }`	`-Infinity`
`{ $atanh: Infinity }` o `{ $atanh: -Infinity }`	Genera un errore.

Quando l'input è `null` o il campo di riferimento è mancante, il risultato è. `null` Un input di `NaN` produce`NaN`. I valori limite `1` e `-1` producono `Infinity` e `-Infinity` rispettivamente. Qualsiasi valore al di fuori dell'`[-1, 1]`intervallo (incluso`±Infinity`) causa un errore.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$atanh`operatore per calcolare la tangente iperbolica inversa di un valore.

Crea documenti di esempio

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 0.5 },
  { "_id": 3, "value": -0.5 }
]);
```

Esempio di interrogazione

```
db.values.aggregate([
  { $project: {
    "result": { $atanh: "$value" }
  }}
]);
```

Output

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 0.5493061443340548 },
  { "_id": 3, "result": -0.5493061443340548 }
```

```
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$atanh` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

L'operatore di `$avg` aggregazione in Amazon DocumentDB calcola il valore medio dell'espressione specificata tra i documenti che vengono inseriti nello stage. Questo operatore è utile per calcolare la media di un campo numerico o di un'espressione in un set di documenti.

Parametri

- `expression`: L'espressione da utilizzare per calcolare la media. Può essere un percorso di campo (ad es.`"$field"`) o un'espressione (ad es.`{ $multiply: ["$field1", "$field2"] }`).

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare l'`$avg`operatore per calcolare il punteggio medio in una serie di documenti degli studenti.

Crea documenti di esempio

```
db.students.insertMany([
  { name: "John", score: 85 },
  { name: "Jane", score: 92 },
  { name: "Bob", score: 78 },
  { name: "Alice", score: 90 }
]);
```

Esempio di interrogazione

```
db.students.aggregate([
  { $group: {
    _id: null,
    avgScore: { $avg: "$score" }
```

```
  }}
]);
```

Output

```
[
  {
    "_id": null,
    "avgScore": 86.25
  }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$avg`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce la media dell'espressione specificata per i documenti in ogni finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

Create documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`city`, ordina ogni partizione in base `day` e restituisce una media corrente del valore `temp` compreso tra l'inizio della partizione e il documento corrente.

```
db.readings.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$city",
      sortBy: { day: 1 },
```

```
      output: {
        runningAvg: {
          $avg: "$temp",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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 }
]
```

Ogni documento è `runningAvg` arricchito dalla media dei contenuti `temp` all'interno della partizione fino al documento corrente incluso.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$avg` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano sia l'utilizzo dell'accumulatore (in`$group`) che l'utilizzo dell'operatore di finestra (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()
```

$Dimensione binaria

Nuovo dalla versione 8.0.1.

L'`$binarySize`operatore in Amazon DocumentDB restituisce la dimensione in byte di una determinata stringa o valore di dati binari. Per i valori di stringa, si tratta del conteggio dei byte UTF-8 codificati, non del conteggio dei caratteri. Multi-byte UTF-8 i caratteri (come i caratteri accentati o i caratteri CJK) produrranno un numero di byte maggiore del numero di caratteri nella stringa.

Parametri

- `expression`: un'espressione che si risolve in una stringa o in un valore di dati binari.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$binarySize`operatore per restituire la dimensione in byte dei campi stringa.

Crea documenti di esempio

```
db.docs.insertMany([
  {_id: 1, content: "hello"},
  {_id: 2, content: "Amazon DocumentDB"},
  {_id: 3, content: ""}
]);
```

Esempio di interrogazione

```
db.docs.aggregate([
  { $project: { size: { $binarySize: "$content" } } }
```

```
]);
```

Output

```
[
  {_id: 1, size: 5},
  {_id: 2, size: 17},
  {_id: 3, size: 0}
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$binarySize` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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 e

Nuovo dalla versione 8.0.1.

L'`$bitAnd`operatore in Amazon DocumentDB esegue un'operazione AND bit per bit su valori interi o lunghi.

Parametri

- `expressions`: Una matrice di due o più espressioni, che può essere risolta in numeri interi o lunghi.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$bitAnd`operatore per eseguire AND bit per bit su due campi.

Crea documenti di esempio

```
db.flags.insertMany([
  {_id: 1, a: 13, b: 10},
  {_id: 2, a: 7, b: 5},
  {_id: 3, a: 15, b: 9}
]);
```

Esempio di interrogazione

```
db.flags.aggregate([
  { $project: { result: { $bitAnd: ["$a", "$b"] } } }
]);
```

Output

```
[
  {_id: 1, result: 8},
  {_id: 2, result: 5},
  {_id: 3, result: 9}
]
```

In binario: 13 (1101) E 10 (1010) = 8 (1000); 7 (0111) E 5 (0101) = 5 (0101); 15 (1111) E 9 (1001) = 9 (1001).

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$bitAnd` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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()
```

$bitNot

Nuovo dalla versione 8.0.1.

L'`$bitNot`operatore in Amazon DocumentDB esegue un'operazione NOT bit per bit su un valore intero o lungo, restituendo il complemento bit per bit. Per un dato numero intero o lungo n, il risultato è - (n+1).

Parametri

- `expression`: Un'espressione che si risolve in un numero intero o lungo. Restituisce il complemento bit per bit, che è - (n+1) per un dato numero intero o lungo n.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$bitNot`operatore per calcolare il complemento bit per bit di valori interi.

Crea documenti di esempio

```
db.numbers.insertMany([
```

```
  {_id: 1, value: 0},
  {_id: 2, value: 5},
  {_id: 3, value: -3}
]);
```

Esempio di interrogazione

```
db.numbers.aggregate([
  { $project: { result: { $bitNot: "$value" } } }
]);
```

Output

```
[
  {_id: 1, result: -1},
  {_id: 2, result: -6},
  {_id: 3, result: 2}
]
```

In binario: NON 0 = -1; NON 5 = -6; NON -3 = 2.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$bitNot` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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 o

Nuovo dalla versione 8.0.1.

L'`$bitOr`operatore in Amazon DocumentDB esegue un'operazione OR bit per bit su valori interi o lunghi.

Parametri

- `expressions`: Una matrice di due o più espressioni, che può essere risolta in numeri interi o lunghi.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$bitOr`operatore per eseguire OR bit per bit su due campi.

Crea documenti di esempio

```
db.flags.insertMany([
  {_id: 1, a: 13, b: 10},
  {_id: 2, a: 7, b: 5},
  {_id: 3, a: 15, b: 9}
]);
```

Esempio di interrogazione

```
db.flags.aggregate([
  { $project: { result: { $bitOr: ["$a", "$b"] } } }
]);
```

Output

```
[
  {_id: 1, result: 15},
  {_id: 2, result: 7},
  {_id: 3, result: 15}
]
```

In binario: 13 (1101) O 10 (1010) = 15 (1111); 7 (0111) O 5 (0101) = 7 (0111); 15 (1111) O 9 (1001) = 15 (1111).

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'`$bitOr`operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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()
```

$BitXor

Nuovo dalla versione 8.0.1.

L'`$bitXor`operatore in Amazon DocumentDB esegue un'operazione XOR bit per bit su valori interi o lunghi.

Parametri

- `expressions`: Una matrice di due o più espressioni, che può essere risolta in numeri interi o lunghi.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$bitXor`operatore per eseguire XOR bit per bit su due campi.

Crea documenti di esempio

```
db.flags.insertMany([
  {_id: 1, a: 13, b: 10},
  {_id: 2, a: 7, b: 5},
  {_id: 3, a: 15, b: 9}
]);
```

Esempio di interrogazione

```
db.flags.aggregate([
  { $project: { result: { $bitXor: ["$a", "$b"] } } }
]);
```

Output

```
[
  {_id: 1, result: 7},
  {_id: 2, result: 2},
  {_id: 3, result: 6}
]
```

In binario: 13 (1101) XOR 10 (1010) = 7 (0111); 7 (0111) XOR 5 (0101) = 2 (0010); 15 (1111) XOR 9 (1001) = 6 (0110).

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$bitXor` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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()
```

$fondo

Nuovo dalla versione 8.0.1.

Usa l'`$bottom`accumulatore nella `$group` fase per restituire il documento con il punteggio più basso per gruppo in base a un ordinamento specificato.

Parametri

- `sortBy`: Un documento che specifica l'ordinamento. `1`Da utilizzare in ordine crescente o `-1` decrescente.
- `output`: un'espressione che specifica i campi da restituire dalla parte inferiore del documento.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$bottom`accumulatore per trovare la vendita più bassa (quantità minima) per articolo in una raccolta di vendite.

Crea documenti di esempio

```
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 }
])
```

Esempio di interrogazione

```
db.sales.aggregate([
  { $group: { _id: "$item", bottomSale: { $bottom: { sortBy: { quantity: -1 }, output:
 { quantity: "$quantity", price: "$price" } } } } }
])
```

Output

```
[
  { "_id": "xyz", "bottomSale": { "quantity": 7, "price": 6 } },
  { "_id": "abc", "bottomSale": { "quantity": 5, "price": 8 } }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$bottom`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce il documento `output` dall'ultimo posto (in base a quello dell'operatore`sortBy`) tra i documenti in ciascuna finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un documento. `window`

> **Note**
>
> Se utilizzato come operatore di finestra in`$setWindowFields`, `$bottom` è limitato a 100 MB di dati intermedi. Un'operazione che supera questo limite restituisce un errore.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`player`, ordina ogni partizione in base `round` e restituisce il punteggio più basso visualizzato dall'inizio della partizione fino al documento corrente.

```
db.matchScores.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$player",
      sortBy: { round: 1 },
      output: {
        worstRoundSoFar: {
          $bottom: { sortBy: { score: -1 }, output: { round: "$round", score:
 "$score" } },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
```

```
]);
```

Output

```
[
  { "_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 } }
]
```

Ogni documento è arricchito con il punteggio più basso all'`worstRoundSoFar`interno della relativa partizione fino al documento corrente incluso.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'`$bottom`operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano sia l'utilizzo dell'accumulatore (in`$group`) che l'utilizzo dell'operatore di finestra (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()
```

$ in basso N

Nuovo dalla versione 8.0.1.

Usa l'`$bottomN`accumulatore nella `$group` fase per restituire gli ultimi N elementi all'interno di un gruppo in base a un ordinamento specificato. Se un gruppo contiene meno di N elementi, `$bottomN` restituisce tutti gli elementi del gruppo.

Parametri

- n: un numero intero positivo o un'espressione che si risolve in uno, che specifica il numero di risultati finali da restituire per gruppo.
- `sortBy`: un documento che specifica il criterio di ordinamento. `1`Da utilizzare in ordine crescente o `-1` decrescente.
- `output`: un'espressione che specifica i campi da restituire da ciascuno degli ultimi N documenti.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$bottomN`accumulatore per trovare le ultime 2 vendite (quantità minima) per articolo in una raccolta vendite.

Crea documenti di esempio

```
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 }
])
```

Esempio di interrogazione

```
db.sales.aggregate([
  { $group: { _id: "$item", bottomTwoSales: { $bottomN: { n: 2, sortBy: { quantity:
 -1 }, output: { quantity: "$quantity", price: "$price" } } } } }
])
```

Output

```
[
  { "_id": "xyz", "bottomTwoSales": [{ "quantity": 9, "price": 6 }, { "quantity": 3,
 "price": 12 }] },
  { "_id": "abc", "bottomTwoSales": [{ "quantity": 7, "price": 8 }, { "quantity": 5,
 "price": 10 }] }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$bottomN`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce un array dei seguenti `output` N documenti (in base a quelli dell'operatore`sortBy`) tra i documenti di ciascuna finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

> **Note**
>
> Se utilizzato come operatore di finestra in`$setWindowFields`, `$bottomN` è limitato a 100 MB di dati intermedi. Un'operazione che supera questo limite restituisce un errore.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`player`, ordina ogni partizione in base `round` e restituisce i due round con il punteggio più basso visualizzati dall'inizio della partizione fino al documento corrente.

```
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"] }
        }
      }
    }
  }
]);
```

Output

```
[
```

```
  { "_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 }] }
]
```

Ogni documento è arricchito dai due round con `bottomTwoRoundsSoFar` il punteggio più basso all'interno della relativa partizione fino al documento corrente incluso. Se la finestra contiene meno di due documenti, vengono restituiti tutti.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$bottomN` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano sia l'utilizzo dell'accumulatore (in`$group`) che l'utilizzo dell'operatore di finestra (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()
```

$bsonSize

Nuovo dalla versione 8.0.1.

L'`$bsonSize`operatore in Amazon DocumentDB restituisce la dimensione in byte di un documento se codificato come BSON.

Parametri

- `expression`: Un'espressione che si risolve in un documento o oggetto.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$bsonSize`operatore per restituire la dimensione BSON di ogni documento.

Crea documenti di esempio

```
db.items.insertMany([
  {_id: 1, name: "a", tags: ["x"]},
  {_id: 2, name: "ab", tags: ["x", "y", "z"]},
  {_id: 3, name: "abc", tags: []}
]);
```

Esempio di interrogazione

```
db.items.aggregate([
  { $project: { docSize: { $bsonSize: "$$ROOT" } } }
]);
```

Output

```
[
  {_id: 1, docSize: 46},
  {_id: 2, docSize: 65},
  {_id: 3, docSize: 39}
]
```

> **Note**
>
> Le dimensioni esatte dipendono dalla codifica BSON dei campi del documento.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$bsonSize` dell'operatore, scegli la scheda relativa alla lingua che desideri utilizzare:

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

Novità dalla versione 8.0

Non supportato dal cluster Elastic.

La fase di `$bucket` aggregazione in Amazon DocumentDB consente di raggruppare i documenti di input in bucket in base a un'espressione e ai limiti del bucket specificati. Ciò può essere utile per analizzare dati che rientrano in determinati intervalli di valori o categorie.

Parametri

- `groupBy`(obbligatorio): L'espressione che specifica il valore in base al quale raggruppare.
- `boundaries`(obbligatorio): Una matrice di valori doppi che definiscono i limiti del bucket. I documenti vengono assegnati ai bucket in base al valore dell'`groupBy`espressione che rientra nei limiti specificati.
- `default`(opzionale): Un valore letterale che viene emesso per i documenti il cui valore di `groupBy` espressione non rientra in nessuno dei limiti specificati.
- `output`(opzionale): Un oggetto che specifica le informazioni da restituire per ogni bucket. È possibile utilizzare operatori di accumulazione come`$sum`, `$avg``$min`, e `$max` per calcolare aggregazioni per ogni bucket.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare lo `$bucket` stage per raggruppare i dati di vendita per fascia di prezzo.

Crea documenti di esempio

```
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") }
]);
```

Esempio di query

```
db.sales.aggregate([
  {
    $bucket: {
      groupBy: "$price",
      boundaries: [0, 10, 20, 30],
      default: "Other",
      output: {
        "count": { $sum: 1 },
        "totalQuantity": { $sum: "$quantity" }
      }
    }
  },
```

```
  {
    $sort: { _id: 1 }
  }
])
```

Output

```
[
  { _id: 0, count: 1, totalQuantity: 3 },
  { _id: 10, count: 2, totalQuantity: 4 },
  { _id: 20, count: 2, totalQuantity: 2 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$bucket` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

Nuovo dalla versione 8.0.2.

La fase di `$bucketAuto` aggregazione in Amazon DocumentDB raggruppa automaticamente i documenti di input in un numero specificato di bucket in base a un'espressione specificata. A differenza `$bucket` di quanto avviene quando si definiscono esplicitamente i limiti dei bucket, i limiti `$bucketAuto` vengono determinati automaticamente in modo che i documenti siano distribuiti nel modo più uniforme possibile sul numero di bucket richiesto. Ciò può essere utile quando si desidera classificare i dati in un numero fisso di gruppi senza conoscere in anticipo la distribuzione dei valori.

Parametri

- `groupBy`(obbligatorio): l'espressione che specifica il valore in base al quale raggruppare.
- `buckets`(obbligatorio): un numero intero positivo che specifica il numero di bucket in cui sono raggruppati i documenti. A seconda dei documenti di input e dell'`groupBy`espressione, il numero effettivo di bucket può essere inferiore al numero specificato.
- `output`(opzionale): un oggetto che specifica le informazioni da emettere per ogni bucket. È possibile utilizzare operatori di accumulo come`$sum`, `$avg``$min`, e `$max` per calcolare le aggregazioni per ogni bucket. Se omesso, ogni bucket genera un campo per impostazione predefinita. `count`
- `granularity`(opzionale): una stringa che specifica la serie numerica preferita da utilizzare per i limiti del bucket. Questo aggancia i limiti calcolati a una progressione numerica standard. Questo parametro è disponibile solo se tutti i `groupBy` valori sono numerici e nessuno lo è. `NaN` I valori validi sono: `R5` `R10`,`R20`,`R40`,`R80`,`1-2-5`,`E6`,`E12`,`E24`,,`E48`, `E96` `E192`, `ePOWERSOF2`.

Output

Ogni bucket viene restituito come documento separato che contiene un `_id` oggetto che descrive i limiti del bucket:

- `_id.min`: Il limite inferiore incluso del bucket.
- `_id.max`: Il limite superiore del secchio. Questo limite è esclusivo per ogni bucket tranne l'ultimo bucket della serie, dove è incluso.

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare la `$bucketAuto` fase per raggruppare i dati di vendita in bucket in base al prezzo.

Crea documenti di esempio

```
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") }
]);
```

Esempio di interrogazione

Il comando seguente raggruppa i documenti di esempio in tre fasce in base al prezzo:

```
db.sales.aggregate([
  {
    $bucketAuto: {
      groupBy: "$price",
      buckets: 3,
      output: {
        "count": { $sum: 1 },
        "totalQuantity": { $sum: "$quantity" }
      }
    }
  }
])
```

Output

Questo comando restituisce il seguente output:

```
[
  { _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 }
]
```

Esempio di interrogazione con granularità

L'esempio seguente aggiunge l'`granularity`opzione per agganciare i limiti del bucket calcolati a potenze di 2.

```
db.sales.aggregate([
  {
    $bucketAuto: {
      groupBy: "$price",
      buckets: 2,
      granularity: "POWERSOF2",
      output: {
        "count": { $sum: 1 },
        "totalQuantity": { $sum: "$quantity" }
      }
    }
  }
])
```

Output

Questo comando restituisce il seguente output:

```
[
  { _id: { min: 4, max: 16 }, count: 3, totalQuantity: 7 },
  { _id: { min: 16, max: 32 }, count: 2, totalQuantity: 2 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$bucketAuto` comando, scegliete la scheda corrispondente alla lingua che desiderate utilizzare:

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

Novità dalla versione 4.0

L'`$ceil`operatore in Amazon DocumentDB, come in MongoDB, arrotonda un numero al numero intero più vicino. Ciò è utile quando è necessario eseguire operazioni matematiche su campi numerici e garantire che il risultato sia un numero intero.

Parametri

- `expression`: L'espressione numerica da arrotondare.

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'`$ceil`operatore per arrotondare un campo numerico.

Crea documenti di esempio

```
db.numbers.insertMany([
  { "_id": 1, "value": 3.14 },
  { "_id": 2, "value": -2.7 },
  { "_id": 3, "value": 0 }
])
```

Esempio di query

```
db.numbers.aggregate([
  { $project: {
    "roundedUp": { $ceil: "$value" }
  }}
])
```

Output

```
{ "_id": 1, "roundedUp": 4 }
{ "_id": 2, "roundedUp": -2 }
{ "_id": 3, "roundedUp": 0 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$ceil` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$Cambia stream

Non supportato da Elastic Cluster.

La fase di `$changeStream` aggregazione apre un cursore del flusso di modifiche per monitorare le modifiche in tempo reale a una raccolta. Restituisce i documenti relativi agli eventi di modifica quando si verificano operazioni di inserimento, aggiornamento, sostituzione o eliminazione.

Parametri

- `fullDocument`: specifica se restituire il documento completo per le operazioni di aggiornamento. Le opzioni sono `default` o `updateLookup`.
- `resumeAfter`: facoltativo. Riprendi il token per continuare da un punto specifico del flusso di modifiche.
- `startAtOperationTime`: facoltativo. Timestamp da cui iniziare il flusso di modifiche.
- `allChangesForCluster`: facoltativo. Valore booleano. Quando`true`, controlla tutte le modifiche nel cluster (per il database di amministrazione). Quando `false` (impostazione predefinita), guarda solo la raccolta specificata.
- `showExpandedEvents`: facoltativo. Valore booleano. Quando`true`, include le notifiche di eventi DDL (Expanded Data Definition Language), ad esempio `createCollection` `ecreateIndexes`, nel flusso di modifiche. Quando `false` (impostazione predefinita), questi eventi non vengono restituiti. Disponibile in Amazon DocumentDB 8.0 a partire dalla versione del motore 8.0.2.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dello `$changeStream` stage per monitorare le modifiche a una raccolta.

Esempio di interrogazione

```
// 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()));
}
```

Output

```
{
  _id: { _data: '...' },
  operationType: 'insert',
  clusterTime: Timestamp(1, 1234567890),
  fullDocument: { _id: 1, item: 'Widget', qty: 10 },
  ns: { db: 'test', coll: 'inventory' },
  documentKey: { _id: 1 }
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo della fase di `$changeStream` aggregazione, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

$modifica StreamSplitLargeEvent

Questa fase di aggregazione non è supportata dai cluster elastici.

Questa fase è disponibile a partire dalla versione 8.0.2 del motore Amazon DocumentDB.

La fase di `$changeStreamSplitLargeEvent` aggregazione viene utilizzata all'interno di una pipeline $ChangeStream per suddividere gli eventi superiori a 16 MB in frammenti più piccoli. Restituisce ogni frammento in sequenza tramite il cursore del flusso di modifica.

Parametri

Nessuna. L'input per lo `$changeStreamSplitLargeEvent` stage deve essere un documento vuoto.

Esempio: MongoDB Shell

L'esempio seguente dimostra l'utilizzo dello `$changeStreamSplitLargeEvent` stage per suddividere un evento di modifica di grandi dimensioni in più frammenti.

Esempio di interrogazione

```
// 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()));
}
```

Output

```
{
  _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: [] }
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo della fase di `$changeStreamSplitLargeEvent` aggregazione, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$cmp`operatore in Amazon DocumentDB viene utilizzato per confrontare due valori e restituire un valore intero che ne indica l'ordine relativo. È un operatore di confronto che confronta due espressioni e restituisce un valore intero pari a -1, 0 o 1, a seconda che il primo valore sia rispettivamente minore, uguale o maggiore del secondo valore.

Parametri

- `expression1`: la prima espressione da confrontare.
- `expression2`: La seconda espressione da confrontare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$cmp`operatore per confrontare due valori numerici.

Crea documenti di esempio

```
db.collection.insertMany([
  { _id: 1, value1: 10, value2: 20 },
  { _id: 2, value1: 15, value2: 15 },
  { _id: 3, value1: 20, value2: 10 }
]);
```

Esempio di interrogazione

```
db.collection.find({
  $expr: {
    $cmp: ["$value1", "$value2"]
  }
})
```

Output

```
[
  { "_id" : 1, "value1" : 10, "value2" : 20 },
  { "_id" : 3, "value1" : 20, "value2" : 10 }
]
```

In questo esempio, l'`$cmp`operatore confronta i `value2` campi `value1` e per ogni documento. Il risultato è:

```
- `$cmp: ["$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).
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$cmp` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

Node.js

Ecco un esempio di utilizzo dell'`$cmp`operatore in un' Node.js applicazione con il `mongodb` driver:

```
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

Ecco un esempio di utilizzo dell'`$cmp`operatore in un'applicazione Python con il `pymongo` driver:

```
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()
```

L'output degli esempi in Python Node.js e in Python sarà lo stesso dell'esempio di MongoDB Shell:

```
[
  { "_id" : 1, "value1" : 10, "value2" : 20 },
  { "_id" : 2, "value1" : 15, "value2" : 15 },
  { "_id" : 3, "value1" : 20, "value2" : 10 }
]
```

$collStats

Novità dalla versione 4.0

La fase di `$collStats` aggregazione in Amazon DocumentDB fornisce statistiche sulla raccolta specificata, in modo simile al comando `db.collection.stats()` nella shell MongoDB. Questa fase può essere utilizzata per recuperare informazioni sulla raccolta, come il numero di documenti, la dimensione totale della raccolta e vari parametri prestazionali.

Parametri

- `latencyStats`: (opzionale) Un documento che specifica le opzioni per la raccolta delle statistiche sulla latenza. Questo parametro non è supportato in Amazon DocumentDB.
- `recordStats`: (opzionale) Un documento che specifica le opzioni per la raccolta di statistiche sui record. Questo parametro non è supportato in Amazon DocumentDB.

- `queryExecStats`: (opzionale) Un documento che specifica le opzioni per la raccolta delle statistiche sull'esecuzione delle query. Questo parametro non è supportato in Amazon DocumentDB.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare la fase di `$collStats` aggregazione per recuperare statistiche su una raccolta denominata nel database. `test` db

Crea documenti di esempio

```
db.test.insertMany([
  { "name": "John", "age": 30 },
  { "name": "Jane", "age": 25 },
  { "name": "Bob", "age": 35 }
]);
```

Esempio di interrogazione

```
db.test.aggregate([
  { $collStats: {} }
]);
```

Output

```
{
  "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)
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$collStats` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

Node.js

Ecco un esempio di come utilizzare la fase di `$collStats` aggregazione in un' Node.js applicazione che utilizza il driver ufficiale MongoDB Node.js :

```
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

Ecco un esempio di come utilizzare la fase di `$collStats` aggregazione in un'applicazione Python utilizzando PyMongo il driver:

```
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

L'operatore di `$concat` aggregazione in Amazon DocumentDB concatena (o combina) più stringhe in un documento per produrre una singola stringa che può essere restituita all'applicazione. Ciò riduce il lavoro svolto nell'applicazione, poiché la manipolazione delle stringhe viene eseguita a livello di database.

Parametri

- `expression1`: la prima stringa da concatenare.
- `expression2`: La seconda stringa da concatenare.
- `...`: stringhe aggiuntive da concatenare (opzionale).

Esempio (MongoDB Shell)

In questo esempio, concateniamo il nome e il cognome degli utenti per produrre il nome completo di ogni persona.

Crea documenti di esempio

```
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"}
])
```

Esempio di interrogazione

```
db.people.aggregate([
  { $project: { full_name: { $concat: [ "$first_name", " ", "$last_name"] } } }
])
```

Output

```
{ "_id" : 1, "full_name" : "Jane Doe" }
{ "_id" : 2, "full_name" : "John Doe" }
{ "_id" : 3, "full_name" : "Steve Smith" }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$concat` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'operatore di `$concatArrays` aggregazione in Amazon DocumentDB viene utilizzato per concatenare due o più array in un unico array. Questo può essere utile quando è necessario combinare più array di dati in un unico array per ulteriori elaborazioni o analisi.

Parametri

- `array1`: il primo array da concatenare.

- `array2`: Il secondo array da concatenare.
- `[array3, ...]`: (opzionale) Matrici aggiuntive da concatenare.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$concatArrays`operatore per combinare due array in un unico array.

Crea documenti di esempio

```
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"]
  }
]);
```

Esempio di interrogazione

```
db.collection.aggregate([
  {
    $project: {
      _id: 0,
      name: 1,
      all_activities: { $concatArrays: ["$hobbies", "$skills"] }
    }
  }
]);
```

Output

```
[
  {
```

```
    "name": "John Doe",
    "all_activities": [
      "reading",
      "swimming",
      "programming",
      "design"
    ]
  },
  {
    "name": "Jane Smith",
    "all_activities": [
      "hiking",
      "cooking",
      "marketing",
      "analysis"
    ]
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$concatArrays` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$cond`operatore in Amazon DocumentDB viene utilizzato per valutare un'espressione condizionale e restituire una delle due possibili espressioni di risultato.

Parametri

- `if`: L'espressione booleana da valutare.

- `then`: L'espressione da restituire se l'`if`espressione è vera.
- `else`: L'espressione da restituire se l'`if`espressione è falsa.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'uso dell'`$cond`operatore per restituire un valore basato sull'età di una persona.

Crea documenti di esempio

```
db.people.insertMany([
  { _id: 1, name: "John Doe", age: 35 },
  { _id: 2, name: "Jane Doe", age: 25 },
  { _id: 3, name: "Bob Smith", age: 65 }
]);
```

Esempio di interrogazione

```
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"
            }
          }
        }
      }
    }
  }
])
```

Output

```
[
  { "_id" : 1, "name" : "John Doe", "ageGroup" : "middle-aged" },
  { "_id" : 2, "name" : "Jane Doe", "ageGroup" : "young" },
  { "_id" : 3, "name" : "Bob Smith", "ageGroup" : "elderly" }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del $cond comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

$converti

Nuovo dalla versione 4.0

L'`$convert`operatore in Amazon DocumentDB viene utilizzato per convertire un valore da un tipo di dati a un altro. Questo operatore è utile quando è necessario eseguire operazioni su dati di diverso tipo, come convertire una stringa in un numero o una data in un timestamp.

Parametri

- `to`: il tipo di dati di destinazione in cui convertire il valore. I valori supportati sono `"string"`, `"double"`, `"long"`, `"int"`, `"date"` e `"boolean"`.
- `from`: il tipo di dati corrente del valore. Se non specificato, Amazon DocumentDB tenterà di rilevare automaticamente il tipo di dati.
- `onError`: (opzionale) Il valore da restituire se la conversione fallisce. Può essere un valore specifico o uno dei seguenti valori speciali:`"null"`,`"zerofill"`, o`"error"`.
- `onNull`: (opzionale) Il valore da restituire se il valore di input è`null`. Può essere un valore specifico o uno dei seguenti valori speciali:`"null"`,`"zerofill"`, o`"error"`.

Esempio (MongoDB Shell)

L'esempio seguente mostra la conversione di un valore di stringa in una data utilizzando l'operatore. `$convert`

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.users.aggregate([
  {
    $project: {
      _id: 1,
      name: 1,
```

```
      joinedOn: {
        $convert: {
          input: "$joinedOn",
          to: "date",
          onError: "null",
          onNull: "null"
        }
      }
    }
  }
])
```

Output

```
[
  { "_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 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$convert` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

Nuovo dalla versione 8.0.1.

L'`$cos`operatore in Amazon DocumentDB restituisce il coseno di un valore misurato in radianti. Utilizzalo nella pipeline di aggregazione per eseguire calcoli trigonometrici su campi numerici.

Parametri

- `expression`: Un'espressione che si risolve in un numero in radianti.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$cos`operatore per calcolare il coseno dei valori angolari in radianti.

Crea documenti di esempio

```
db.angles.insertMany([
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 1.0471975511965976 },
  { "_id": 3, "angle": 3.141592653589793 }
]);
```

Esempio di interrogazione

```
db.angles.aggregate([
  { $project: {
    "cosine": { $cos: "$angle" }
  }}
]);
```

Output

```
[
  { "_id": 1, "cosine": 1 },
  { "_id": 2, "cosine": 0.5000000000000001 },
  { "_id": 3, "cosine": -1 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo $cos dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

Nuovo dalla versione 8.0.1.

L'`$cosh`operatore in Amazon DocumentDB restituisce il coseno iperbolico di un valore. Utilizzalo nella pipeline di aggregazione per eseguire calcoli trigonometrici iperbolici su campi numerici.

Parametri

- `expression`: Un'espressione che si risolve in un numero.

Il tipo restituito è predefinito`double`. Se l'input è un decimale a 128 bit, anche l'output è un decimale a 128 bit.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$cosh`operatore per calcolare il coseno iperbolico di valori numerici.

Crea documenti di esempio

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Esempio di interrogazione

```
db.values.aggregate([
  { $project: {
    "result": { $cosh: "$value" }
  }}
]);
```

Output

```
[
  { "_id": 1, "result": 1 },
  { "_id": 2, "result": 1.5430806348152437 },
  { "_id": 3, "result": 1.5430806348152437 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$cosh` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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 (fase)

La fase di `$count` aggregazione in Amazon DocumentDB viene utilizzata per contare il numero di documenti che passano alla fase. Viene spesso utilizzata come fase finale di una pipeline di aggregazione per restituire il numero totale di documenti corrispondenti alle fasi precedenti.

Parametri

- `field`: Il campo da contare. Questo parametro è facoltativo e, se non fornito, lo stage conterà il numero totale di documenti di input.

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare lo `$count` stage per ottenere il numero totale di documenti in una raccolta.

Crea documenti di esempio

```
db.users.insertMany([
  { name: "John", age: 30 },
  { name: "Jane", age: 25 },
  { name: "Bob", age: 35 },
  { name: "Alice", age: 28 }
]);
```

Esempio di interrogazione

```
db.users.aggregate([
  { $count: "total" }
]);
```

Output

```
{ "total" : 4 }
```

L'esempio aggrega la `users` raccolta e utilizza la `$count` fase per contare il numero totale di documenti.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$count` comando, scegliete la scheda corrispondente alla lingua che desiderate utilizzare:

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 (accumulatore)

Nuovo dalla versione 8.0.1.

Utilizza l'`$count`accumulatore in una `$group` fase per restituire il numero di documenti in ciascun gruppo. Accetta un oggetto vuoto `{}` come argomento.

Questa è diversa dalla fase della `$count` pipeline, che è una fase autonoma che conta tutti i documenti che passano attraverso la pipeline. L'`$count`accumulatore conta invece i documenti all'interno di ciascun gruppo prodotto da. `$group`

Sintassi

```
{ $count: {} }
```

Parametri

- L'`$count`accumulatore non accetta argomenti. Accetta un oggetto vuoto `{}` e restituisce il numero di documenti in ogni gruppo.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$count`accumulatore per contare il numero di prodotti in ciascuna categoria.

Crea documenti di esempio

```
db.products.insertMany([
  { name: "Widget", category: "A" },
  { name: "Gadget", category: "A" },
  { name: "Doohickey", category: "B" },
  { name: "Thingamajig", category: "B" },
  { name: "Whatsit", category: "B" }
])
```

Esempio di interrogazione

```
db.products.aggregate([
  { $group: { _id: "$category", count: { $count: {} } } }
])
```

Output

```
[
  { "_id": "A", "count": 2 },
  { "_id": "B", "count": 3 }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$count`accumulatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce il numero di documenti in ogni finestra. Si

specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

Create documenti di esempio

```
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 }
])
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`category`, ordina ogni partizione in base `day` e restituisce un numero progressivo di documenti dall'inizio della partizione fino al documento corrente.

```
db.events.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$category",
      sortBy: { day: 1 },
      output: {
        runningCount: {
          $count: {},
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
])
```

Output

```
[
  { "_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 }
]
```

A ogni documento viene aggiunto il numero di documenti all'interno della relativa partizione fino al documento corrente incluso. `runningCount`

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'`$count`accumulatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano sia l'utilizzo dell'accumulatore (in`$group`) che l'utilizzo dell'operatore di finestra (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()
```

$covarianza POP

Nuovo dalla versione 8.0.2.

L'`$covariancePop`operatore in Amazon DocumentDB restituisce la covarianza della popolazione di due espressioni numeriche. È un operatore finestra che viene utilizzato solo nella `$setWindowFields` fase; non è valido nelle fasi o. `$group` `$bucket` Si specifica l'operatore `output` nel campo dello stage e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

Parametri

- `$covariancePop`utilizza una matrice a due elementi`[ <expression1>, <expression2> ]`, in cui ogni elemento è un'espressione numerica (un percorso di campo come `"$x"` un'espressione o una costante). Restituisce la covarianza della popolazione delle due espressioni nei documenti nella finestra.

Esempio (MongoDB Shell)

L'esempio seguente partiziona i documenti in base`series`, ordina ogni partizione in base `t` e calcola la covarianza della popolazione di e sull'intera partizione. `x` `y`

Create documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.measurements.aggregate([
```

```
  {
    $setWindowFields: {
      partitionBy: "$series",
      sortBy: { t: 1 },
      output: {
        covariance: {
          $covariancePop: [ "$x", "$y" ],
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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 }
]
```

Ogni documento in una partizione riceve la stessa `covariance` covarianza della popolazione di `x` e `y` tra tutti i documenti di quella partizione: `2` per le serie `A` (x e y aumentano insieme) e `-2` per le serie `B` (y diminuisce all'aumentare di x).

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$covariancePop` dell'operatore nella `$setWindowFields` fase, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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()
```

$covarianzeAMP

Nuovo dalla versione 8.0.2.

L'`$covarianceSamp`operatore in Amazon DocumentDB restituisce la covarianza campionaria di due espressioni numeriche. È un operatore finestra che viene utilizzato solo nella `$setWindowFields` fase; non è valido nelle fasi o. `$group` `$bucket` Si specifica l'operatore `output` nel campo dello stage e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

Parametri

- `$covarianceSamp`utilizza una matrice a due elementi`[ <expression1>, <expression2> ]`, in cui ogni elemento è un'espressione numerica (un percorso di campo come `"$x"` un'espressione o una costante). Restituisce la covarianza campionaria delle due espressioni nei documenti nella finestra.

Esempio (MongoDB Shell)

L'esempio seguente partiziona i documenti in base`series`, ordina ogni partizione in base `t` e calcola la covarianza campionaria di e sull'intera partizione. `x` `y`

Create documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.measurements.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$series",
      sortBy: { t: 1 },
      output: {
        covariance: {
          $covarianceSamp: [ "$x", "$y" ],
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
```

```
    }
  }
]);
```

Output

```
[
  { "_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 }
]
```

Ogni documento in una partizione riceve la stessa `covariance` covarianza campionaria di `x` e `y` tra tutti i documenti di quella partizione: `3` per le serie `A` (`x` e `y` aumentano insieme) e `-3` per le serie `B` (`y` diminuisce all'aumentare di `x`). La covarianza del campione si divide per `n - 1` anzichén, quindi è maggiore in termini di ampiezza rispetto alla covarianza della popolazione per gli stessi dati.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'`$covarianceSam`operatore nella `$setWindowFields` fase, scegliete la scheda corrispondente alla lingua che desiderate utilizzare:

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

La fase `$currentOp` di aggregazione restituisce informazioni sulle operazioni attualmente in esecuzione nel database. Questa fase è utile per monitorare le interrogazioni e le operazioni attive in una pipeline di aggregazione.

Parametri

- `allUsers`(opzionale): se impostato su`true`, restituisce le operazioni per tutti gli utenti. Il valore predefinito è “`false`”.
- `idleConnections`(opzionale): se impostato su`true`, include le connessioni inattive. Il valore predefinito è “`false`”.

- `idleCursors`(opzionale): Se impostato su`true`, include informazioni sui cursori inattivi. Il valore predefinito è “`false`”.
- `idleSessions`(opzionale): se impostato su`true`, include informazioni sulle sessioni inattive. Il valore predefinito è “`true`”.
- `localOps`(opzionale): se impostato su`true`, include le operazioni locali. Il valore predefinito è “`false`”.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo della fase di `$currentOp` aggregazione per recuperare informazioni sulle operazioni di lettura attive.

Esempio di interrogazione

```
db.aggregate([
  { $currentOp: { allUsers: true, idleConnections: false } },
  { $match: { op: "query" } }
])
```

Output

```
[
  {
    "opid": "12345",
    "active": true,
    "op": "query",
    "ns": "test.users",
    "secs_running": 2
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$currentOp` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$dataAggiungi

Nuovo dalla versione 5.0

L'operatore di `$dateAdd` aggregazione in Amazon DocumentDB consente di aggiungere una durata a un valore di data e ora.

Parametri

- `date`: Un valore di data e ora a cui aggiungere una durata.
- `duration`: La durata da aggiungere al `date` valore. Può essere specificato come oggetto con le chiavi per`years`,`months`, `weeksdays`,`hours`,`minutes`, `eseconds`.
- `timezone`: (opzionale) Il fuso orario da utilizzare quando si esegue l'aggiunta della data. Se non specificato, viene utilizzato il fuso orario predefinito del cluster Amazon DocumentDB.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$dateAdd`operatore per aggiungere 2 giorni e 12 ore a una data.

Crea documenti di esempio

```
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") }
]);
```

Esempio di interrogazione

```
db.events.aggregate([
  {
    $project: {
      _id: 1,
      eventDate: 1,
      eventDatePlustwodaysandtwelvehours: {
        $dateAdd: {
          startDate: {
            $dateAdd: {
              startDate: "$eventDate",
              unit: "day",
              amount: 2
            }
          },
          unit: "hour",
```

```
          amount: 12
        }
      }
    }
  }
])
```

Output

```
[
  {
    "_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)"
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$dateAdd` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

Nuovo dalla versione 5.0

Non supportato dal cluster Elastic.

L'operatore `$dateDiff` di aggregazione calcola la differenza tra due date in unità specificate. Restituisce il numero di limiti di unità superati tra le date di inizio e di fine.

Parametri

- `startDate`: L'espressione della data di inizio.
- `endDate`: L'espressione della data di fine.
- `unit`: L'unità di tempo per la differenza. Le unità supportate sono `year` `quarter`,`month`,`week`,`day`,`hour`,`minute`,`second`, e`millisecond`.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$dateDiff`operatore per calcolare il numero di giorni tra l'immissione dell'ordine e la consegna.

Crea documenti di esempio

```
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")
  }
]);
```

Esempio di interrogazione

```
db.shipments.aggregate([
  {
    $project: {
      orderId: 1,
      orderDate: 1,
      deliveryDate: 1,
      daysToDeliver: {
        $dateDiff: {
          startDate: "$orderDate",
          endDate: "$deliveryDate",
          unit: "day"
        }
      }
    }
  }
]);
```

Output

```
[
  {
```

```
    _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
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$dateDiff` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$data FromParts

Novità dalla versione 5.0.1 e 8.0

L'operatore di `$dateFromParts` aggregazione in Amazon DocumentDB costruisce e restituisce un valore di data e ora dalle sue parti costitutive, come anno, mese, giorno, ora, minuto, secondo e millisecondo.

Parametri

- `year`: L'anno solare. Richiesto quando non si utilizzano parti ISO 8601.
- `month`,`day`,,`hour`, `minutesecond`,`millisecond`: (opzionale) I restanti componenti di data e ora. Per impostazione predefinita, i componenti omessi utilizzano il valore valido più basso.
- `isoWeekYear`,`isoWeek`,`isoDayOfWeek`: (opzionale) Componenti della data ISO 8601. Utilizzali al posto di `yearmonth`, e `day` per costruire una data a partire da una data settimanale ISO 8601.
- `timezone`: (opzionale) Il fuso orario da utilizzare per costruire la data. Se non specificato, viene utilizzato l'UTC.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$dateFromParts`operatore per costruire una data a partire da singoli componenti.

Crea documenti di esempio

```
db.parts.insertMany([
  { _id: 1, y: 2023, m: 4, d: 1 },
  { _id: 2, y: 2023, m: 12, d: 25 }
]);
```

Esempio di interrogazione

```
db.parts.aggregate([
  {
    $project: {
      _id: 1,
      constructedDate: {
        $dateFromParts: { year: "$y", month: "$m", day: "$d" }
      }
```

```
    }
  }
])
```

Output

```
[
  { "_id": 1, "constructedDate": ISODate("2023-04-01T00:00:00Z") },
  { "_id": 2, "constructedDate": ISODate("2023-12-25T00:00:00Z") }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$dateFromParts` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$data FromString

L'operatore di `$dateFromString` aggregazione in Amazon DocumentDB consente di convertire una stringa data-ora in un oggetto data. Ciò è utile quando i dati sono archiviati come stringhe, ma è necessario eseguire operazioni basate sulla data sui dati.

Parametri

- `dateString`: Una stringa che rappresenta una data e un'ora.
- `format`: (opzionale) Una stringa che specifica il formato di. `dateString` Se non viene fornito, Amazon DocumentDB tenterà di analizzare la stringa nel formato. ISO-8601

- `timezone`: (opzionale) Una stringa che specifica il fuso orario. Se non viene fornito, Amazon DocumentDB utilizzerà il fuso orario del server.
- `onError`: (opzionale) Speciifica l'azione da intraprendere se la conversione fallisce. I valori possibili sono `'error'` (impostazione predefinita, che genera un errore), `'null'` (restituiscenull) o `'replace'` (sostituisce il valore con la stringa di sostituzione specificata nell'onErrorMessageopzione).
- `onErrorMessage`: (opzionale) Se `onError` è impostata su`'replace'`, questa opzione specifica la stringa sostitutiva.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare per `$dateFromString` convertire una stringa di data in un oggetto data in Amazon DocumentDB.

Crea documenti di esempio

```
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"}
]);
```

Esempio di interrogazione

```
db.missionLog.aggregate([
  {
    $project: {
      event: '$event',
      logDate: {
        $dateFromString: {
          dateString: '$logDate'
        }
      }
    }
  }
]);
```

Output

```
[
  {
    "_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")
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$dateFromString` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$dateSottrai

Nuovo dalla versione 5.0

L'operatore di `$dateSubtract` aggregazione in Amazon DocumentDB consente di sottrarre una durata specificata da un valore di data.

Parametri

- `date`: espressione di data che si risolve in una data o un timestamp.
- `subtrahend`: espressione di durata che specifica la quantità di tempo da sottrarre dall'espressione. `date`
- `unit`: Una stringa che specifica l'unità di tempo per l'espressione. `subtrahend` Le unità supportate sono «anno», «trimestre», «mese», «settimana», «giorno», «ora», «minuto», «secondo» e «millisecondo».

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$dateSubtract`operatore per calcolare la data di un anno fa dalla data corrente.

Crea documenti di esempio

```
db.events.insertOne({
  eventName: "Player joined",
  eventTime: ISODate("2023-04-01T12:00:00Z")
});
```

Esempio di query

```
db.events.aggregate([
  {
    $project: {
```

```
      eventName: 1,
      oneYearAgo: {
        $dateSubtract: {
          startDate: "$eventTime",
          amount: 1,
          unit: "year"
        }
      }
    }
  }
])
```

Output

```
{
  "_id" : ObjectId("64567890abcdef012345678"),
  "eventName" : "Player joined",
  "oneYearAgo" : ISODate("2022-04-01T12:00:00Z")
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$dateSubtract` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$data ToParts

Novità dalla versione 5.0.1 e 8.0

L'operatore di `$dateToParts` aggregazione in Amazon DocumentDB restituisce un documento che contiene le parti costitutive di un valore di data e ora, come anno, mese, giorno, ora, minuto, secondo e millisecondo.

Parametri

- `date`: Il valore di data e ora da suddividere in parti.
- `timezone`: (opzionale) Il fuso orario da utilizzare per l'estrazione delle parti della data. Se non specificato, le parti vengono restituite in formato UTC.
- `iso8601`: (opzionale) Se impostato su`true`, il documento restituito utilizza parti di data ISO 8601 (`isoWeekYear`,`isoWeek`,`isoDayOfWeek`). Il valore predefinito è `false`.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$dateToParts`operatore per restituire le singole parti di una data.

Crea documenti di esempio

```
db.events.insertMany([
  { _id: 1, eventDate: ISODate("2023-04-01T10:30:15Z") },
  { _id: 2, eventDate: ISODate("2023-12-25T18:45:00Z") }
]);
```

Esempio di interrogazione

```
db.events.aggregate([
  {
    $project: {
      _id: 1,
      parts: { $dateToParts: { date: "$eventDate" } }
```

```
    }
  }
])
```

Output

```
[
  {
    "_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 }
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$dateToParts` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$data ToString

L'operatore di `$dateToString` aggregazione in Amazon DocumentDB viene utilizzato per convertire un valore di data o timestamp in una rappresentazione di stringa. Ciò è utile quando è necessario formattare la data e l'ora in un modo specifico per la visualizzazione o l'ulteriore elaborazione.

Parametri

- `date`: Il valore della data o del timestamp da convertire in una stringa.
- `format`: Una stringa che specifica il formato in cui deve essere rappresentata la data. La stringa di formato può includere diversi specificatori di formato, ad esempio %Y per l'anno a quattro cifre, per il mese a due cifre, `%m` per il giorno del mese a due cifre, `%d` ecc.
- `timezone`: (opzionale) Il fuso orario da utilizzare per la conversione. Se non specificato, viene utilizzato il fuso orario del server che ospita il cluster Amazon DocumentDB.
- `onNull`: (opzionale) Il valore da restituire se il `date` parametro è`null`.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$dateToString`operatore per formattare il `logDate` campo della raccolta. `missionLog`

Crea documenti di esempio

```
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") }
]);
```

Esempio di interrogazione

```
db.missionLog.aggregate([
  {
    $project: {
      event: "$event",
      logDateFormatted: {
        $dateToString: {
          format: "%Y-%m-%d %H:%M:%S",
          date: "$logDate"
        }
      }
    }
  }
])
```

Output

```
[
  {
    "_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"
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$dateToString` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

Node.js

Ecco un esempio di utilizzo dell'`$dateToString`operatore in un' Node.js applicazione:

```
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

Ecco un esempio di utilizzo dell'`$dateToString`operatore in un'applicazione 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']

    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

Nuovo dalla versione 8.0

Non supportato dal cluster Elastic.

L'operatore di `$dateTrunc` aggregazione in Amazon DocumentDB tronca una data in un'unità specificata.

Parametri

- `date`: espressione di data che si risolve in una data o un timestamp.
- `unit`: Una stringa che specifica l'unità di tempo per l'espressione sottraend. Le unità supportate sono`year`,`quarter`,`month`,,`week`, `dayhour`, `minute` e. `second` `millisecond`

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$dateTrunc`operatore per troncare una data all'ora.

Crea documenti di esempio

```
db.events.insertMany([
  {
    eventName: "Event 1",
    eventTime: ISODate("2025-04-01T12:15:00Z")
  },
  {
    eventName: "Event 2",
```

```
    eventTime: ISODate("2025-08-15T14:33:22Z")
  },
]);
```

Esempio di interrogazione

```
db.events.aggregate([
  {
    $project: {
      eventName: 1,
      eventTime: 1,
      truncatedToHour: {
        $dateTrunc: {
          date: "$eventTime",
          unit: "hour"
        }
      }
    }
  }
]);
```

Output

```
[
  {
    _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')
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$dateTrunc` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$giorno OfMonth

L'operatore di `$dayOfMonth` aggregazione in Amazon DocumentDB recupera il giorno del mese (da 1 a 31) per una determinata data. Questo operatore è utile per raggruppare, filtrare o estrarre il giorno del mese dai campi della data nei documenti.

Parametri

- `date expression`: L'espressione della data può essere un campo data dal documento, un oggetto data o una stringa di data.

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'`$dayOfMonth`operatore per estrarre il giorno del mese da un campo data nel documento.

Crea documenti di esempio

```
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") }
]);
```

Esempio di query

```
db.events.aggregate([
  { $project: {
    eventDay: { $dayOfMonth: "$eventDate" }
  }}
])
```

Output

```
[
  { "_id" : 1, "eventDay" : 15 },
  { "_id" : 2, "eventDay" : 28 },
  { "_id" : 3, "eventDay" : 1 },
  { "_id" : 4, "eventDay" : 30 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$dayOfMonth` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$giorno OfWeek

L'`$dayOfWeek`operatore in Amazon DocumentDB estrae il giorno della settimana da un determinato campo data. Restituisce il giorno della settimana come numero compreso tra 1 (domenica) e 7 (sabato), che è lo stesso comportamento di MongoDB.

Parametri

- `date field`: Il campo della data da cui estrarre il giorno della settimana.

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'`$dayOfWeek`operatore per estrarre il giorno della settimana dal `date` campo della raccolta. `weather`

Crea documenti di esempio

```
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")
  }
]);
```

Esempio di interrogazione

```
db.weather.aggregate([
  {
    $project: {
      dayOfWeek: { $dayOfWeek: "$date" }
    }
  }
]).pretty();
```

Output

```
{ "_id" : ObjectId("64272c6663f4f8ce422c2d91"), "dayOfWeek" : 7 }
{ "_id" : ObjectId("64272c6663f4f8ce422c2d92"), "dayOfWeek" : 1 }
{ "_id" : ObjectId("64272c6663f4f8ce422c2d93"), "dayOfWeek" : 2 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$dayOfWeek` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$giorno OfYear

L'`$dayOfYear`operatore in Amazon DocumentDB restituisce il giorno dell'anno per una data con un numero compreso tra 1 e 366 (365 negli anni non bisestili).

Parametri

- `expression`: Il campo o l'espressione della data da cui estrarre il giorno dell'anno.

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'`$dayOfYear`operatore per estrarre il giorno dell'anno da un campo data in una raccolta Amazon DocumentDB.

Crea documenti di esempio

```
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")
  }
```

```
])
```

Esempio di interrogazione

```
db.weather.aggregate([
  {
    $project: {
      dayOfYear: { $dayOfYear: "$date" }
    }
  }
]).pretty()
```

Output

```
{ "_id" : ObjectId("642b86fc7d8e07af279bbe63"), "dayOfYear" : 74 }
{ "_id" : ObjectId("642b86fc7d8e07af279bbe64"), "dayOfYear" : 365 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$dayOfYear` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$gradi ToRadians

Nuovo dalla versione 8.0.1.

L'`$degreesToRadians`operatore in Amazon DocumentDB converte un valore da gradi a radianti.

Parametri

- `expression`: Un'espressione che si risolve in un valore numerico in gradi.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$degreesToRadians`operatore per convertire i valori in gradi in radianti.

Create documenti di esempio

```
db.angles.insertMany([
  { "_id": 1, "degrees": 0 },
  { "_id": 2, "degrees": 90 },
  { "_id": 3, "degrees": 180 },
  { "_id": 4, "degrees": 360 }
]);
```

Esempio di interrogazione

```
db.angles.aggregate([
  { $project: {
    "radians": { $degreesToRadians: "$degrees" }
  }}
]);
```

Output

```
[
  { "_id": 1, "radians": 0 },
  { "_id": 2, "radians": 1.5707963267948966 },
  { "_id": 3, "radians": 3.141592653589793 },
  { "_id": 4, "radians": 6.283185307179586 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$degreesToRadians` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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()
```

$DenseRank

Nuovo dalla versione 8.0.2.

L'operatore `$denseRank` window in Amazon DocumentDB restituisce la posizione di ogni documento all'interno della sua partizione, in base all'ordine delle `$setWindowFields` fasi. `sortBy` I documenti con lo stesso `sortBy` valore ricevono la stessa classificazione e il valore distinto successivo utilizza la posizione immediatamente successiva, senza lasciare spazi vuoti (ad esempio,). `1, 2, 2, 3` È specificato nel `output` campo dello `$setWindowFields` stage.

È un operatore di finestra che viene utilizzato solo nello `$setWindowFields` stage.

Parametri

- `$denseRank`accetta `{}` come valore un documento vuoto.
- Non accetta una `window` cornice.
- Richiede esattamente una `sortBy` chiave sul `$setWindowFields` palco.

Esempio (MongoDB Shell)

L'esempio seguente suddivide i documenti in base`game`, ordina ogni partizione in ordine decrescente e `score` assegna a ciascun giocatore un rango denso.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.players.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$game",
```

```
      sortBy: { score: -1 },
      output: {
        denseRank: { $denseRank: {} }
      }
    }
  }
]);
```

Output

```
[
  { "_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 }
]
```

All'interno di ogni `game` partizione, i giocatori sono classificati in ordine decrescente`score`. Bob e Carol sono in parità `85` e condividono la classifica`2`, e Dave segue in classifica `3` senza alcun divario.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$denseRank` dell'operatore nella `$setWindowFields` fase, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

L'`$divide`operatore nella pipeline di aggregazione di Amazon DocumentDB viene utilizzato per dividere un numero per un altro. È un operatore utile per eseguire operazioni matematiche su campi numerici all'interno dei documenti.

Parametri

- `numerator`: Il dividendo o il numero da dividere.
- `denominator`: Il divisore o il numero per cui dividere.

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'`$divide`operatore per calcolare la tariffa oraria per i dipendenti in base al loro stipendio annuale e al numero di ore lavorative all'anno.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.employees.aggregate([
  {
    $project: {
      name: 1,
      hourlyRate: { $divide: ["$salary", "$hoursPerYear"] }
    }
  }
])
```

Output

```
[
  { "_id" : 1, "name" : "John Doe", "hourlyRate" : 38.46153846153846 },
  { "_id" : 2, "name" : "Jane Smith", "hourlyRate" : 43.26923076923077 },
  { "_id" : 3, "name" : "Bob Johnson", "hourlyRate" : 36.05769230769231 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$divide` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$Numero del documento

Nuovo dalla versione 8.0.2.

L'operatore `$documentNumber` window in Amazon DocumentDB restituisce un numero sequenziale distinto per ogni documento nella relativa partizione, in base all'ordine delle fasi. `$setWindowFields` `sortBy` A differenza di `$rank` and`$denseRank`, non tratta i legami in modo speciale: ogni documento riceve un numero univoco (ad esempio`1, 2, 3, 4`) e i documenti con `sortBy` valori uguali vengono numerati nell'ordine in cui vengono rilevati. È specificato nel `output` campo dello `$setWindowFields` stage.

È un operatore di finestra che viene utilizzato solo nello `$setWindowFields` stage.

Parametri

- `$documentNumber`accetta `{}` come valore un documento vuoto.
- Non accetta una `window` cornice.
- Richiede esattamente una `sortBy` chiave sul `$setWindowFields` palco.

Esempio (MongoDB Shell)

L'esempio seguente suddivide i documenti in base`game`, ordina ogni partizione in ordine decrescente e `score` assegna a ciascun giocatore un numero di documento sequenziale.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.players.aggregate([
  {
    $setWindowFields: {
```

```
      partitionBy: "$game",
      sortBy: { score: -1 },
      output: {
        documentNumber: { $documentNumber: {} }
      }
    }
  }
]);
```

Output

```
[
  { "_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 }
]
```

All'interno di ogni `game` partizione, ogni giocatore riceve un numero sequenziale distinto in ordine decrescente`score`. Bob e Carol sono pari 85 ma sono comunque numerati 2 e 3 nell'ordine in cui si incontrano.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$documentNumber` dell'operatore nella `$setWindowFields` fase, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

L'operatore di `$eq` aggregazione confronta due valori e restituisce `true` se sono uguali, altrimenti restituisce. `false`

Parametri

- `expression1`: Il primo valore da confrontare.
- `expression2`: Il secondo valore da confrontare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo `$eq` dell'operatore per verificare se le quantità di prodotto corrispondono ai valori target.

Crea documenti di esempio

```
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 }
]);
```

Esempio di query

```
db.inventory.aggregate([
  {
    $project: {
      item: 1,
      qty: 1,
      target: 1,
      meetsTarget: { $eq: ["$qty", "$target"] }
    }
  }
]);
```

Output

```
[
  { _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 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di `$eq` aggregazione, scegli la scheda relativa alla lingua che desideri utilizzare:

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

Novità dalla versione 4.0

L'`$exp`operatore in Amazon DocumentDB consente di elevare la costante e a un determinato numero.

Parametri

- `expression`: L'espressione da valutare. Può trattarsi di qualsiasi espressione di aggregazione valida, inclusi riferimenti di campo, operazioni aritmetiche e altre fasi di aggregazione.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'uso dell'`$exp`operatore per trovare tutti i documenti in cui il `quantity` campo è maggiore del campo. `price`

Crea documenti di esempio

```
db.items.insertMany([
  { item: "canvas", quantity: 4 },
  { item: "journal", quantity: 2 }
]);
```

Esempio di interrogazione

```
db.items.aggregate([
  { $project: {
    "quantityRaised": {$exp: "$quantity"}}
  }
]);
```

Output

```
[
  {
```

```
    _id: ObjectId('6920b785311cf98b79d2950d'),
    quantityRaised: 54.598150033144236
  },
  {
    _id: ObjectId('6920b785311cf98b79d2950e'),
    quantityRaised: 7.38905609893065
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$exp` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

Nuovo dalla versione 8.0.2.

La fase di `$facet` aggregazione in Amazon DocumentDB elabora più pipeline di aggregazione in un'unica fase sullo stesso set di documenti di input. Ogni campo di output definisce la propria pipeline e ogni pipeline riceve gli stessi documenti di input. Ciò consente di calcolare diverse aggregazioni indipendenti, come conteggi, raggruppamenti e statistiche di riepilogo, in una singola query invece di eseguire la raccolta tramite più query separate.

Ogni sottopipeline viene eseguita in modo indipendente e produce una serie di documenti archiviati nel campo di output corrispondente. Poiché le pipeline secondarie non condividono lo stato, l'output di una pipeline non influisce su un'altra.

Sintassi

```
{
  $facet: {
    <outputField1>: [ <stage1>, <stage2>, ... ],
    <outputField2>: [ <stage1>, <stage2>, ... ],
    ...
  }
}
```

Parametri

- `outputField`(obbligatorio): il nome di un campo di output che contiene l'array di risultati della pipeline di aggregazione associata. È possibile specificare uno o più campi di output. Ogni campo di output è mappato a una serie di fasi della pipeline di aggregazione che viene applicata ai documenti di input.

Output

Lo `$facet` stage restituisce un singolo documento. Ogni campo di output specificato contiene una serie di documenti prodotti dalla corrispondente sottopipeline.

Limitazioni

- Una `$facet` pipeline secondaria non può includere le seguenti fasi:`$changeStream`,`$collStats`, `$facet` (la nidificazione non è consentita),,,,`$geoNear`,`$indexStats`,`$listSearchIndexes`, `$merge``$out`, `$planCacheStats` e. `$search` `$searchMeta` `$vectorSearch`
- Il documento restituito `$facet` dallo stage deve rimanere entro il limite di dimensione del documento BSON di 16 MB.
- Una `$sort` fase che precede una `$facet` fase non può determinare l'ordine di un accumulatore sensibile all'ordine (`$first`,,,`$last`, `$push` `$firstN``$lastN`, o`$mergeObjects`) in una`$group`,`$bucket`, o `$bucketAuto` fase in cui tale ordinamento dovrebbe essere applicato oltre i confini. `$facet` Ciò vale sia che la fase di raggruppamento si trovi all'interno di una `$facet` sottopipeline o dopo la fase. `$facet` In questo caso, Amazon DocumentDB restituisce un errore invece di eseguire la pipeline. Per ordinare un accumulatore di questo tipo, posiziona lo `$sort` stage all'interno della `$facet` sottopipeline, immediatamente prima della fase di raggruppamento. Order-insensitive gli accumulatori, quali,, e `$sum` `$min` `$max``$count`, non sono interessati.

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare lo `$facet` stage per calcolare diverse aggregazioni sugli stessi dati di vendita in una singola query.

Crea documenti di esempio

```
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") }
]);
```

Esempio di interrogazione

Il comando seguente viene utilizzato `$facet` per calcolare, in un'unica fase, la quantità totale venduta per articolo, le statistiche complessive dei prezzi e il numero totale di documenti:

```
db.sales.aggregate([
  {
    $facet: {
      "totalByItem": [
        { $group: { _id: "$item", totalQuantity: { $sum: "$quantity" } } }
      ],
      "priceStats": [
        { $group: { _id: null, avgPrice: { $avg: "$price" }, maxPrice: { $max:
 "$price" } } }
      ],
      "documentCount": [
        { $count: "total" }
      ]
    }
  }
])
```

Output

Questo comando restituisce il seguente output:

```
[
  {
    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 }
    ]
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$facet` comando, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

L'`$filter`operatore in Amazon DocumentDB viene utilizzato per applicare un'espressione di filtro a ciascun elemento di un array e restituire un array contenente solo gli elementi che corrispondono

alla condizione specificata. Questo operatore è utile quando devi eseguire complesse operazioni di filtraggio sui campi dell'array all'interno dei tuoi documenti.

Parametri

- `input`: il campo dell'array da filtrare.
- `as`: il nome della variabile da usare per ogni elemento dell'`input`array all'interno dell'`cond`espressione.
- `cond`: L'espressione booleana che determina se un determinato elemento deve essere incluso nell'array di output.

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare l'`$filter`operatore per proiettare il cliente di ogni ordine e creare un nuovo campo di matrice PaidItems contenente solo gli articoli dell'array items in cui il prezzo è maggiore di 15. In sostanza, filtra gli articoli di ogni ordine per includere solo i prodotti che costano più di 15€.

Crea documenti di esempio

```
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 }
  ]}
]);
```

Esempio di interrogazione

```
db.orders.aggregate([
  {
    $project: {
```

```
      customer: 1,
      paidItems: {
        $filter: {
          input: "$items",
          as: "item",
          cond: { $gt: ["$$item.price", 15] }
        }
      }
    }
  }
]).pretty();
```

Output

```
[
  {
    _id: 1,
    customer: 'abc123',
    paidItems: [ { name: 'Product B', price: 20, qty: 1 } ]
  },
  { _id: 2, customer: 'def456', paidItems: [] },
  { _id: 3, customer: 'ghi789', paidItems: [] }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$filter` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

Nuovo dalla versione 5.0.

Non supportato da Elastic Cluster.

L'`$first`operatore in Amazon DocumentDB restituisce il primo documento da un insieme raggruppato di documenti. Viene comunemente utilizzato nelle pipeline di aggregazione per recuperare il primo documento che corrisponde a una condizione specifica.

Parametri

- `expression`: L'espressione da restituire come primo valore in ogni gruppo.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'uso dell'`$first`operatore per recuperare il valore del primo elemento riscontrato per ciascuna categoria durante l'aggregazione.

Nota: `$first` restituisce il primo documento in base all'ordine corrente dei documenti nella pipeline. Per garantire un ordine specifico (ad esempio, per data, prezzo, ecc.), è necessario utilizzare una `$sort` fase prima della `$group` fase.

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.products.aggregate([
  { $group: { _id: "$category", firstItem: { $first: "$item" } } }
]);
```

Output

```
[
  { "_id" : "food", "firstItem" : "abc" },
  { "_id" : "toy", "firstItem" : "xyz" }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$first`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce il valore `expression` del primo documento in ogni finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

> **Note**
>
> Se utilizzato come operatore di finestra in`$setWindowFields`, `$first` è limitato a 100 MB di dati intermedi. Un'operazione che supera questo limite restituisce un errore.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`ticker`, ordina ogni partizione in base `hour` e restituisce il primo (il primo) prezzo registrato in ciascuna partizione.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        firstPrice: {
          $first: "$price",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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 }
]
```

A ogni documento viene aggiunto il `firstPrice` prezzo del primo documento nella relativa finestra (il primo nella partizione). `hour`

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$first` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano sia l'utilizzo dell'accumulatore (in`$group`) che l'utilizzo dell'operatore di finestra (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()
```

$primoN

Nuovo dalla versione 8.0.1.

L'`$firstN`operatore in Amazon DocumentDB restituisce i primi N elementi. Quando viene utilizzato come accumulatore in una `$group` fase, restituisce un array dei primi N valori in ogni gruppo. Se utilizzato come operatore di espressione di matrice, restituisce i primi N elementi di un array.

Parametri

- `input`: L'espressione che si risolve nel campo o nell'array da cui restituire i valori.
- n: un numero intero positivo che specifica quanti valori restituire. Se utilizzato come accumulatore in uno `$group` stage, n può anche essere un'espressione, purché si risolva in un numero intero positivo in base al campo del gruppo. `_id`

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$firstN`accumulatore per recuperare le prime due quantità per ogni articolo durante l'aggregazione.

> **Note**
>
> `$firstN`seleziona i valori nell'ordine in cui i documenti arrivano alla fase. `$group` Per restituire i primi N valori per un ordine specifico (ad esempio, per data o punteggio), aggiungi una `$sort` fase precedente. `$group`

Crea documenti di esempio

```
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") }
```

```
]);
```

Esempio di interrogazione

```
db.sales.aggregate([
  { $group: { _id: "$item", firstTwoQuantities: { $firstN: { input: "$quantity", n:
 2 } } } }
]);
```

Output

```
[
  { "_id": "abc", "firstTwoQuantities": [10, 5] },
  { "_id": "xyz", "firstTwoQuantities": [15, 7] }
]
```

Esempio di utilizzo delle espressioni (MongoDB Shell)

L'`$firstN`operatore può essere utilizzato anche come espressione all'interno di uno `$project` stage per restituire i primi N elementi di un campo di matrice.

Crea documenti di esempio

```
db.inventory.insertMany([
  { _id: 1, item: "abc", tags: ["red", "green", "blue", "yellow", "purple"] },
  { _id: 2, item: "xyz", tags: ["alpha", "beta", "gamma"] }
]);
```

Esempio di interrogazione

```
db.inventory.aggregate([
  { $project: {
      firstThreeTags: { $firstN: { input: "$tags", n: 3 } }
    }}
]);
```

Output

```
[
  { "_id": 1, "firstThreeTags": ["red", "green", "blue"] },
  { "_id": 2, "firstThreeTags": ["alpha", "beta", "gamma"] }
```

```
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$firstN`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce un array contenente fino ai primi n valori per i documenti in ogni finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

> **Note**
>
> Se utilizzato come operatore di finestra in`$setWindowFields`, `$firstN` è limitato a 100 MB di dati intermedi. Un'operazione che supera questo limite restituisce un errore.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`ticker`, ordina ogni partizione in base `hour` e restituisce i primi due prezzi visualizzati dall'inizio della partizione fino al documento corrente.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        firstTwoSoFar: {
          $firstN: { input: "$price", n: 2 },
```

```
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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] }
]
```

Ogni documento è arricchito con `firstTwoSoFar` i primi due `price` valori all'interno della relativa finestra (in ordine). `hour`

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$firstN` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano l'uso dell'accumulatore (in`$group`), l'uso delle espressioni (in`$project`) e l'uso dell'operatore di finestra (in`$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

Novità dalla versione 4.0.

L'`$floor`operatore in Amazon DocumentDB restituisce il numero intero più grande inferiore o uguale al numero specificato. Questo operatore è utile per arrotondare i valori numerici.

Parametri

- `expression`: L'espressione numerica da arrotondare per difetto.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'uso dell'`$floor`operatore per arrotondare un valore decimale al numero intero più vicino.

Crea documenti di esempio

```
db.numbers.insertOne({ value: 3.14 });
```

Esempio di interrogazione

```
db.numbers.aggregate([
  { $project: { _id: 0, floored: { $floor: "$value" } } }
]);
```

Output

```
{ "floored" : 3 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$floor` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

La fase di `$geoNear` aggregazione restituisce i documenti in ordine di prossimità a un punto specificato. Calcola la distanza dal punto e include la distanza nei documenti di output.

Parametri

- `near`: il punto da cui calcolare le distanze, specificato come coordinate GeoJSON o legacy.
- `distanceField`: Il nome del campo per memorizzare la distanza calcolata.
- `spherical`: valore booleano che indica se utilizzare la geometria sferica (obbligatorio per i punti GeoJSON).
- `maxDistance`: facoltativo. Distanza massima dal punto centrale.
- `minDistance`: facoltativo. Distanza minima dal punto centrale.
- `query`: facoltativo. Criteri di filtro aggiuntivi da applicare.
- `limit`: facoltativo. Numero massimo di documenti da restituire.
- `key`: facoltativo. Campo da utilizzare per l'interrogazione geospaziale quando esistono più indici geospaziali.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo `$geoNear` dello stage per trovare i negozi più vicini a una determinata posizione.

Create documenti di esempio

```
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] } }
]);
```

Esempio di interrogazione

```
db.stores.aggregate([
  {
    $geoNear: {
      near: { type: "Point", coordinates: [-122.4, 37.8] },
      distanceField: "distance",
      spherical: true
    }
  }
]);
```

Output

```
[
  { _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 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo della fase di aggregazione `$geoNear`, scegli la scheda relativa alla lingua che desideri utilizzare:

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

Nuovo dalla versione 8.0.2.

La fase di `$graphLookup` aggregazione in Amazon DocumentDB esegue una ricerca ricorsiva su una raccolta. Per ogni documento di input, `$graphLookup` parte da uno o più valori e abbina ripetutamente un campo di connessione a un campo di destinazione, seguendo la catena di documenti corrispondenti fino a quando non ci sono più corrispondenze o viene raggiunta una profondità di ricorsione opzionale. Ciò è utile per attraversare dati gerarchici o simili a grafici, come strutture di reporting organizzativo, alberi di categorie o relazioni di rete.

Parametri

- `from`: la raccolta di destinazione da cercare in modo ricorsivo. La `from` raccolta deve trovarsi nello stesso database.
- `startWith`: L'espressione che specifica il valore o i valori con cui iniziare la ricerca ricorsiva. Questo valore viene confrontato con quello `connectToField` dei documenti della raccolta. `from` Se `startWith` restituisce un array, la ricerca viene eseguita da tutti gli elementi dell'array contemporaneamente.
- `connectFromField`: il nome del campo il cui valore viene utilizzato per connettersi al documento successivo nella ricerca ricorsiva. Se il valore è un array, ogni elemento viene seguito singolarmente durante l'attraversamento.
- `connectToField`: il nome del campo nella `from` raccolta che viene confrontato con il valore di. `connectFromField`

- `as`: il nome del campo dell'array aggiunto a ciascun documento di output. Questo campo contiene i documenti trovati durante la ricerca ricorsiva. Per informazioni sull'ordine di questi documenti, vedere la voce Ordinamento dei risultati nella seguente sezione Comportamento.
- `maxDepth`: un numero intero non negativo che specifica il numero massimo di iterazioni ricorsive da eseguire dopo la ricerca iniziale. Questo parametro è facoltativo. Un valore di `0` esegue solo la corrispondenza iniziale e nessuna ricorsione. Se si omette`maxDepth`, la ricerca continua in modo ricorsivo fino a quando non ci sono più documenti corrispondenti.
- `depthField`: il nome del campo da aggiungere a ciascun documento dell'`as`array che contiene la profondità di ricorsione alla quale il documento è stato trovato. Questo parametro è facoltativo. La profondità è rappresentata come a `NumberLong` e parte da zero, quindi la ricerca iniziale corrisponde a una profondità di. `0`
- `restrictSearchWithMatch`: Un documento che specifica un'ulteriore condizione di corrispondenza applicata ai documenti della ricerca. Questo parametro è facoltativo. La condizione utilizza la stessa sintassi di interrogazione di una `$match` fase e non può includere espressioni di aggregazione.

> **Note**
>
> Lo `$graphLookup` stage è soggetto a un limite di memoria di 100 MB e non supporta. `allowDiskUse:true` Un'operazione che supera questo limite restituisce un errore. Se l'aggregazione include altre fasi, l'`allowDiskUse:true`opzione rimane valida per tali fasi.

Comportamento

- Cross-collection attraversamento: imposta la raccolta `from` in cui desideri cercare. Può trattarsi della stessa raccolta su cui viene eseguita la pipeline o di una raccolta diversa nello stesso database.
- Ordinamento dei risultati: i documenti dell'`as`array non vengono restituiti in alcun ordine garantito.

Esempio (MongoDB Shell)

L'esempio seguente viene utilizzato `$graphLookup` per attraversare una gerarchia di rendicontazione dei dipendenti e, per ogni dipendente, restituire l'intera catena di manager che li precede.

I seguenti comandi, eseguiti nella shell mongosh, inseriscono documenti di esempio per i dipendenti che stabiliscono una gerarchia di reporting:

```
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" }
]);
```

La seguente aggregazione restituisce l'intera catena di gestione per ogni dipendente in un nuovo array: `reportingHierarchy`

```
db.employees.aggregate([
  {
    $graphLookup: {
      from: "employees",
      startWith: "$reportsTo",
      connectFromField: "reportsTo",
      connectToField: "name",
      as: "reportingHierarchy",
      depthField: "level"
    }
  }
]);
```

Questa operazione restituisce il seguente risultato:

```
[
  {
    _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: []
  }
]
```

Ogni documento del dipendente ottiene un `reportingHierarchy` array che contiene i documenti del manager abbinati in modo ricorsivo. Il `level` campo registra la profondità di ricorsione alla quale è stato trovato ciascun manager. Asya non ha alcun `reportsTo` valore, quindi il suo `reportingHierarchy` array è vuoto.

Esempio con MaxDepth e restrict SearchWithMatch (MongoDB Shell)

L'esempio seguente attraversa una rete di connessioni tra persone e utilizza `maxDepth` per limitare la ricorsione e includere solo i documenti che soddisfano una `restrictSearchWithMatch`

condizione aggiuntiva. La ricerca trova le connessioni di una persona iniziale, con una profondità massima di due livelli, mantenendo solo le persone il cui array contiene. `hobbies` `golf`

I seguenti comandi, eseguiti nella shell mongosh, inseriscono documenti di esempio in cui ogni persona `friends` punta in avanti attraverso una catena di connessioni unidirezionale:

```
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" ] }
]);
```

La seguente aggregazione parte da Tanya, attraversa la sua rete di connessioni fino a due livelli di profondità e mantiene solo le persone inserite nell'elenco per hobby: `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 } }
]);
```

Questa operazione restituisce il seguente risultato:

```
[
  {
    _id: 1,
    name: 'Tanya',
    golfConnections: [
```

```
      { name: 'Carole' },
      { name: 'Joseph' },
      { name: 'Angelo' }
    ]
  }
]
```

`golf`Sono incluse solo le connessioni il cui `hobbies` array contiene e l'attraversamento si interrompe dopo due livelli di ricorsione. Shirley è esclusa perché non figura nell'elenco`golf`, e Miguel è escluso perché si trova a tre livelli da Tanya, oltre il limite del. `maxDepth` 2 Poiché le connessioni formano una catena unidirezionale che non rimanda mai a Tanya, il documento iniziale non corrisponde durante la ricerca ricorsiva.

Esempi di codice

Per visualizzare un esempio di codice che crea la gerarchia dei report per i dipendenti a partire dal primo esempio utilizzando un driver MongoDB, scegli la scheda relativa alla lingua che desideri utilizzare:

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

La fase di `$group` aggregazione in Amazon DocumentDB consente di raggruppare i documenti in base a un'espressione specificata ed eseguire varie operazioni cumulative sui dati raggruppati. Questo può essere utile per attività come il calcolo di totali, medie o altre statistiche basate sui dati raggruppati.

Parametri

- `_id`: specifica l'espressione in base alla quale devono essere raggruppati i documenti di input. Può essere un nome di campo, un'espressione calcolata o una combinazione di entrambi.
- `accumulator expressions`: (opzionale) Una o più espressioni cumulative da applicare ai dati raggruppati. Queste espressioni utilizzano gli operatori di accumulazione menzionati sopra.

Esempio (MongoDB Shell)

L'esempio seguente raggruppa i clienti in base alla loro città e calcola l'importo totale dell'ordine per ciascuna città.

Crea documenti di esempio

```
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 }] }
]);
```

Esempio di interrogazione

```
db.customers.aggregate([
  {
    $group: {
      _id: "$city",
      totalOrders: { $sum: { $sum: "$orders.amount" } }
    }
  }
]);
```

Output

```
[
  { _id: 'Chicago', totalOrders: 150 },
  { _id: 'Los Angeles', totalOrders: 450 },
  { _id: 'New York', totalOrders: 500 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$group` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'operatore di `$gt` aggregazione confronta due valori e restituisce `true` se il primo valore è maggiore del secondo, altrimenti restituisce. `false`

Parametri

- `expression1`: il primo valore da confrontare.
- `expression2`: Il secondo valore da confrontare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo `$gt` dell'operatore per identificare i prodotti che superano una soglia di prezzo.

Crea documenti di esempio

```
db.products.insertMany([
  { _id: 1, name: "Laptop", price: 1200 },
  { _id: 2, name: "Mouse", price: 25 },
  { _id: 3, name: "Monitor", price: 400 }
]);
```

Esempio di interrogazione

```
db.products.aggregate([
  {
    $project: {
      name: 1,
      price: 1,
      expensive: { $gt: ["$price", 100] }
    }
  }
]);
```

Output

```
[
  { _id: 1, name: 'Laptop', price: 1200, expensive: true },
  { _id: 2, name: 'Mouse', price: 25, expensive: false },
  { _id: 3, name: 'Monitor', price: 400, expensive: true }
```

```
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di `$gt` aggregazione, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'operatore di `$gte` aggregazione confronta due valori e restituisce `true` se il primo valore è maggiore o uguale al secondo, altrimenti restituisce. `false`

Parametri

- `expression1`: Il primo valore da confrontare.
- `expression2`: Il secondo valore da confrontare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo `$gte` dell'operatore per verificare se gli studenti hanno superato un esame.

Crea documenti di esempio

```
db.students.insertMany([
  { _id: 1, name: "Alice", score: 85 },
  { _id: 2, name: "Bob", score: 60 },
  { _id: 3, name: "Charlie", score: 72 }
]);
```

Esempio di interrogazione

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      score: 1,
      passed: { $gte: ["$score", 70] }
    }
  }
]);
```

Output

```
[
  { _id: 1, name: 'Alice', score: 85, passed: true },
  { _id: 2, name: 'Bob', score: 60, passed: false },
  { _id: 3, name: 'Charlie', score: 72, passed: true }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di `$gte` aggregazione, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'$houroperatore estrae il componente orario da un campo di data o timestamp.

Parametri

- `dateExpression`: la data alla quale viene applicato l'operatore. Questo deve risolversi in una data BSON valida (ad esempio, un campo come $createdAt o una data letterale).

Un parametro può anche essere specificato come documento nel seguente formato:

{data:<dateExpression>, fuso orario:} <timezoneExpression>

Ciò consente di applicare operazioni di data in base al fuso orario.

```
- `<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.
```

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'$houroperatore per estrarre il componente ora da un campo data e raggruppare i dati di conseguenza.

Crea documenti di esempio

```
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") }
]);
```

Esempio di interrogazione

```
db.events.aggregate([
  {
    $project: {
      hour: { $hour: "$timestamp" }
    }
  },
  {
    $group: {
      _id: "$hour",
      count: { $sum: 1 }
    }
  },
  {
    $sort: { _id: 1 }
  }
```

```
]);
```

Output

```
[
  { "_id": 8, "count": 1 },
  { "_id": 10, "count": 1 },
  { "_id": 12, "count": 1 },
  { "_id": 16, "count": 1 },
  { "_id": 23, "count": 1 }
]
```

Questa query raggruppa gli eventi in base alla componente oraria del `timestamp` campo e conta il numero di eventi per ogni ora.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$hour` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$ifNull`operatore viene utilizzato per restituire un valore specificato se l'espressione di input restituisce null o undefined. Questo operatore può essere utile in scenari in cui si desidera fornire un valore predefinito o gestire casi. null/undefined

Parametri

- `expression`: L'espressione da valutare.
- `replacement`: Il valore da restituire se <`expression`> restituisce null o undefined.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$ifNull`operatore per fornire un valore predefinito quando il `name` campo è nullo o non definito.

Crea documenti di esempio

```
db.users.insertMany([
  { _id: 1, name: "John" },
  { _id: 2, name: null },
  { _id: 3 }
]);
```

Esempio di interrogazione

```
db.users.aggregate([
  {
    $project: {
      _id: 1,
      name: { $ifNull: ["$name", "No Name"] }
    }
  }
]);
```

Output

```
[
  { "_id": 1, "name": "John" },
  { "_id": 2, "name": "No Name" },
  { "_id": 3, "name": "No Name" }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$ifNull` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'operatore `$in` di aggregazione verifica se un valore specificato esiste all'interno di un array. Restituisce `true` se il valore viene trovato nell'array e in `false` altro modo.

Parametri

- `value`: Il valore da cercare.
- `array`: L'array in cui cercare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$in`operatore per verificare se esiste una competenza specifica nel set di competenze di ciascun dipendente.

Crea documenti di esempio

```
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"] }
]);
```

Esempio di query

```
db.employees.aggregate([
  {
    $project: {
      name: 1,
      hasPython: { $in: ["Python", "$skills"] }
    }
  }
]);
```

Output

```
[
```

```
  { _id: 1, name: 'Sarah', hasPython: true },
  { _id: 2, name: 'Mike', hasPython: false },
  { _id: 3, name: 'Emma', hasPython: true }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di `$in` aggregazione, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

$indice OfArray

L'`$indexOfArray`operatore in Amazon DocumentDB viene utilizzato per trovare l'indice della prima occorrenza di un elemento specificato in un array. Questo operatore restituisce la posizione dell'indice a base zero del primo elemento dell'array che corrisponde al valore specificato. Se il valore non viene trovato, restituisce -1.

Parametri

- `array`: L'array da cercare.
- `value`: Il valore da cercare nell'array.
- `start`: (opzionale) La posizione nell'array da cui iniziare la ricerca. Il valore predefinito è 0.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'OfArray operatore $index per trovare l'indice della prima occorrenza dell'elemento «mango» nell'array «fruits» per ogni documento.

Crea documenti di esempio

```
db.collection.insertMany([
  { _id: 1, fruits: ["apple", "banana", "cherry", "durian"] },
  { _id: 2, fruits: ["mango", "orange", "pineapple"] },
  { _id: 3, fruits: ["kiwi", "lemon", "mango"] }
]);
```

Esempio di query

```
db.collection.aggregate([
  {
    $project: {
      _id: 1,
      fruitIndex: { $indexOfArray: ["$fruits", "mango"] }
    }
  }
]);
```

Output

```
{ "_id" : 1, "fruitIndex" : 1 }
{ "_id" : 2, "fruitIndex" : 0 }
{ "_id" : 3, "fruitIndex" : 2 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$indexOfArray` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$indice OfBytes

L'OfBytes operatore $index in Amazon DocumentDB viene utilizzato per trovare l'indice iniziale di una sottostringa all'interno di una stringa, in base alle posizioni in byte dei caratteri. Questo può essere

utile quando si lavora con dati di testo che possono contenere caratteri multibyte, come quelli presenti negli script non latini.

Parametri

- `string`: La stringa di input da cercare.
- `substring`: La sottostringa da cercare all'interno della stringa di input.
- `[<start>]`: (opzionale) La posizione iniziale (in base zero) della ricerca. Se non viene specificato, la ricerca inizia all'inizio della stringa.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'uso di `$indexOfBytes` per trovare l'indice del primo trattino in un insieme di stringhe che rappresentano le posizioni delle scrivanie.

Crea documenti di esempio

```
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" }
]);
```

Esempio di query

```
db.people.aggregate([
  { $project: { stateLocation: { $indexOfBytes: ["$Desk", "-"] } } }
]);
```

Output

```
{ "_id" : 1, "stateLocation" : 11 }
{ "_id" : 2, "stateLocation" : 6 }
{ "_id" : 3, "stateLocation" : 7 }
{ "_id" : 4, "stateLocation" : 8 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$indexOfBytes` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$indexOfCP`operatore in Amazon DocumentDB viene utilizzato per trovare l'indice, in punti di codice (CP), della prima occorrenza di una sottostringa specificata all'interno di un'espressione stringa. Questo può essere utile quando si analizza ed estrae il contenuto dai campi stringa.

Parametri

- `string expression`: La stringa da cercare.
- `substring`: La sottostringa da cercare.
- `[<start>]`: (opzionale) La posizione in cui iniziare la ricerca (indice a base zero). Il valore predefinito è 0.

Esempio (MongoDB Shell)

In questo esempio, utilizziamo l'`$indexOfCP`operatore per trovare l'indice della prima occorrenza del carattere trattino, nel campo Desk di ogni documento.

Crea documenti di esempio

```
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"}
])
```

Esempio di interrogazione

```
db.people.aggregate([
  { $project: { stateLocation: { $indexOfCP: [ "$Desk", "-"] } } }
])
```

Output

```
{ "_id" : 1, "stateLocation" : 10 }
```

```
{ "_id" : 2, "stateLocation" : 6 }
{ "_id" : 3, "stateLocation" : 7 }
{ "_id" : 4, "stateLocation" : 8 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$indexOfCP` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

La fase di `$indexStats` aggregazione in Amazon DocumentDB fornisce informazioni sull'utilizzo degli indici all'interno di una raccolta. Questo operatore ti consente di monitorare i modelli di accesso degli indici, il che può aiutarti a prendere decisioni informate sulla gestione e l'ottimizzazione degli indici.

Parametri

Nessuno

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$indexStats`operatore per analizzare l'utilizzo dell'indice in una raccolta Amazon DocumentDB.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.grocery.aggregate([
  { $indexStats: {} }
]);
```

Output

```
[
```

```
  {
    "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 questo esempio, l'`$indexStats`operatore mostra che l'accesso all'`_id_`indice è stato effettuato 5 volte e che l'`product_1`indice è stato effettuato 10 volte dall'ultimo ripristino o riavvio del server.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$indexStats` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$isArray`operatore in Amazon DocumentDB viene utilizzato per verificare se un campo in un documento è un array. Questo operatore può essere utile nelle pipeline di aggregazione e nelle espressioni condizionali per gestire campi di tipo array.

Parametri

- `field`: Il percorso del campo per verificare se si tratta di un array.

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'`$isArray`operatore per identificare i documenti in cui il campo «inventario» è un array.

Crea documenti di esempio

```
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}]}
]);
```

Esempio di interrogazione

```
db.videos.aggregate([
  {
    $match: {
      $isArray: "$inventory"
    }
  },
  {
    $project: {
      _id: 1,
      name: 1,
      "inventory.location": 1,
      "inventory.count": 1
    }
  }
]).pretty();
```

Output

```
{
  "_id": 4,
  "name": "Bravemind",
  "inventory": [
    {
      "location": "Mason City",
      "count": 250
    },
    {
      "location": "Des Moines",
      "count": 1000
    },
    {
```

```
      "location": "Ames",
      "count": 500
    }
  ]
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$isArray` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$è il numero

Nuovo dalla versione 8.0.1.

L'`$isNumber`operatore in Amazon DocumentDB restituisce un valore booleano che indica se una determinata espressione si risolve in un tipo numerico (intero, decimale, doppio, lungo). Nota che gli array non sono considerati numerici, anche se tutti gli elementi dell'array sono numeri. Ad esempio, `$isNumber` applicato ai valori `[1, 2, 3]` restituiti. `false`

Parametri

- `expression`: qualsiasi espressione per verificare se si risolve in un tipo numerico.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$isNumber`operatore per verificare se i valori dei campi sono numerici.

Crea documenti di esempio

```
db.data.insertMany([
  {_id: 1, value: 42},
  {_id: 2, value: "hello"},
  {_id: 3, value: 3.14},
  {_id: 4, value: null}
]);
```

Esempio di interrogazione

```
db.data.aggregate([
  { $project: { isNum: { $isNumber: "$value" } } }
]);
```

Output

```
[
  {_id: 1, isNum: true},
  {_id: 2, isNum: false},
  {_id: 3, isNum: true},
  {_id: 4, isNum: false}
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$isNumber` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

L'`$isoDayOfWeek`operatore in Amazon DocumentDB restituisce il giorno ISO della settimana per una data come valore intero. Il sistema di data settimanale ISO definisce ogni settimana che inizia

il lunedì e termina la domenica, dove la settimana 1 è la settimana che contiene il primo giovedì dell'anno.

Parametri

- `expression`: L'espressione della data per la quale restituire il giorno ISO della settimana.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$isoDayOfWeek`operatore per recuperare il giorno ISO della settimana per una serie di documenti relativi agli eventi.

Crea documenti di esempio

```
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") }
]);
```

Esempio di interrogazione

```
db.events.aggregate([
  { $project: {
    _id: 1,
    eventDate: 1,
    isoDayOfWeek: { $isoDayOfWeek: "$eventDate" }
  }}
]);
```

Output

```
[
  { "_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 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$isoDayOfWeek` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$isoWeek`operatore in Amazon DocumentDB restituisce il numero della settimana ISO per una data. Il sistema di data settimanale ISO è un modo per numerare le settimane in un anno, in cui la prima settimana di un nuovo anno è la settimana che contiene il primo giovedì di quell'anno. È diverso dal calendario gregoriano, in cui la prima settimana di un nuovo anno è la settimana che contiene il 1 gennaio.

Parametri

Nessuno

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$isoWeek`operatore per recuperare il numero della settimana ISO per una determinata data.

Crea documenti di esempio

```
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") }
])
```

Esempio di interrogazione

```
db.dates.aggregate([
  {
    $project: {
      _id: 1,
      isoWeek: { $isoWeek: "$date" }
    }
  }
])
```

Output

```
[
  { "_id": 1, "isoWeek": 52 },
  { "_id": 2, "isoWeek": 52 },
  { "_id": 3, "isoWeek": 1 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$isoWeek` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$isoWeekYear`operatore in Amazon DocumentDB restituisce il numero dell'anno settimanale ISO 8601 per una determinata data. Il numero dell'anno settimanale ISO differisce dall'anno

del calendario gregoriano in quanto l'anno della settimana può essere diverso dall'anno solare, specialmente all'inizio e alla fine dell'anno.

Parametri

- `expression`: espressione della data per la quale restituire il numero dell'anno settimanale ISO 8601.

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'`$isoWeekYear`operatore per recuperare l'anno settimanale ISO 8601 per il campo data di ogni documento nella raccolta di eventi.

Crea documenti di esempio

```
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") }
]);
```

Esempio di interrogazione

```
db.events.aggregate([
  { $project: {
    name: 1,
    isoWeekYear: { $isoWeekYear: "$date" }
  }}
]);
```

Output

```
[
  { "_id" : 1, "name" : "Event 1", "isoWeekYear" : 2023 },
  { "_id" : 2, "name" : "Event 2", "isoWeekYear" : 2023 },
  { "_id" : 3, "name" : "Event 3", "isoWeekYear" : 2023 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$isoWeekYear` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$last`operatore in Amazon DocumentDB viene utilizzato per restituire l'ultimo elemento di un array che corrisponde ai criteri di interrogazione. È particolarmente utile per recuperare il più recente o l'ultimo elemento di un array che soddisfa una condizione specifica.

Parametri

- `expression`: L'espressione da abbinare agli elementi dell'array.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'uso dell'`$last`operatore in combinazione con `$filter` per recuperare l'ultimo elemento da un array che soddisfa una condizione specifica (ad esempio, il soggetto è «scienza»).

Crea documenti di esempio

```
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 }
    ]
  }
]);
```

Esempio di interrogazione

```
db.collection.aggregate([
    { $match: { name: "John" } },
    {
      $project: {
        name: 1,
        lastScienceScore: {
          $last: {
            $filter: {
              input: "$scores",
              as: "score",
              cond: { $eq: ["$$score.subject", "science"] }
            }
          }
        }
      }
    }
  ]);
```

Output

```
[
  {
    _id: 1,
    name: 'John',
    lastScienceScore: { subject: 'science', score: 90 }
```

```
  }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$last`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce il valore `expression` dell'ultimo documento in ogni finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

Create documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`ticker`, ordina ogni partizione in base `hour` e restituisce l'ultimo (ultimo) prezzo registrato in ciascuna partizione.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        lastPrice: {
          $last: "$price",
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
```

```
]);
```

Output

```
[
  { "_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 }
]
```

A ogni documento viene aggiunto il `lastPrice` prezzo dell'ultimo documento nella relativa finestra (l'ultimo nella partizione). `hour`

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$last` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano sia l'utilizzo delle espressioni (in`$project`) che l'uso dell'operatore di finestra (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()
```

$ l'ultimo N

Nuovo dalla versione 8.0.1.

L'`$lastN`operatore in Amazon DocumentDB restituisce gli ultimi N elementi. Se utilizzato come accumulatore in una `$group` fase, restituisce un array degli ultimi N valori in ciascun gruppo. Se utilizzato come operatore di espressione di matrice, restituisce gli ultimi N elementi di un array.

Parametri

- `input`: L'espressione che si risolve nel campo o nell'array da cui restituire i valori.
- n: un numero intero positivo che specifica quanti valori restituire. Se utilizzato come accumulatore in uno `$group` stage, n può anche essere un'espressione, purché si risolva in un numero intero positivo in base al campo del gruppo. `_id`

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$lastN`accumulatore per recuperare le ultime due quantità per ogni articolo durante l'aggregazione.

> **Note**
>
> `$lastN`seleziona i valori nell'ordine in cui i documenti arrivano alla fase. `$group` Per restituire gli ultimi N valori per un ordine specifico (ad esempio, per data o punteggio), aggiungi una `$sort` fase precedente. `$group`

Crea documenti di esempio

```
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") }
]);
```

Esempio di interrogazione

```
db.sales.aggregate([
  { $group: { _id: "$item", lastTwoQuantities: { $lastN: { input: "$quantity", n:
 2 } } } }
```

```
]);
```

Output

```
[
  { "_id": "abc", "lastTwoQuantities": [5, 8] },
  { "_id": "xyz", "lastTwoQuantities": [7, 3] }
]
```

Esempio di utilizzo delle espressioni (MongoDB Shell)

L'`$lastN`operatore può essere utilizzato anche come espressione all'interno di uno `$project` stage per restituire gli ultimi N elementi di un campo di matrice.

Crea documenti di esempio

```
db.inventory.insertMany([
  { _id: 1, item: "abc", tags: ["red", "green", "blue", "yellow", "purple"] },
  { _id: 2, item: "xyz", tags: ["alpha", "beta", "gamma"] }
]);
```

Esempio di interrogazione

```
db.inventory.aggregate([
  { $project: {
      lastThreeTags: { $lastN: { input: "$tags", n: 3 } }
    }}
]);
```

Output

```
[
  { "_id": 1, "lastThreeTags": ["blue", "yellow", "purple"] },
  { "_id": 2, "lastThreeTags": ["alpha", "beta", "gamma"] }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$lastN`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce un array contenente fino agli ultimi

n valori per i documenti in ogni finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

> **Note**
>
> Se utilizzato come operatore di finestra in`$setWindowFields`, `$lastN` è limitato a 100 MB di dati intermedi. Un'operazione che supera questo limite restituisce un errore.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`ticker`, ordina ogni partizione in base `hour` e restituisce i due prezzi più recenti visualizzati dall'inizio della partizione fino al documento corrente.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        lastTwoSoFar: {
          $lastN: { input: "$price", n: 2 },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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] }
]
```

Ogni documento è arricchito con `lastTwoSoFar` gli ultimi due `price` valori all'interno della finestra (in ordine). `hour`

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$lastN` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano l'uso dell'accumulatore (in`$group`), l'uso delle espressioni (in`$project`) e l'uso dell'operatore di finestra (in`$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

L'`$let`operatore in Amazon DocumentDB viene utilizzato per associare variabili a valori e utilizzare tali variabili nell'espressione. Consente di definire variabili locali che possono essere utilizzate nelle espressioni successive all'interno della stessa fase della pipeline di aggregazione.

Parametri

- `vars`: un oggetto che definisce le variabili da utilizzare nell'espressione.
- `in`: L'espressione in cui vengono utilizzate le variabili definite nel parametro vars.

Esempio (MongoDB Shell)

Questo esempio dimostra l'utilizzo dell'`$let`operatore per calcolare l'area di un rettangolo.

Crea documenti di esempio

```
db.shapes.insertMany([
  { name: "Rectangle 1", length: 5, width: 3 },
  { name: "Rectangle 2", length: 7, width: 4 },
  { name: "Rectangle 3", length: 6, width: 2 }
]);
```

Esempio di interrogazione

```
db.shapes.aggregate([
  {
    $project: {
      name: 1,
      area: {
        $let: {
          vars: {
            length: "$length",
            width: "$width"
          },
          in: {
            $multiply: ["$$length", "$$width"]
          }
        }
      }
    }
```

```
  }
])
```

Output

```
[
  {
    "_id": ObjectId("6161e5b1a3eba3c7f2960d03"),
    "name": "Rectangle 1",
    "area": 15
  },
  {
    "_id": ObjectId("6161e5b1a3eba3c7f2960d04"),
    "name": "Rectangle 2",
    "area": 28
  },
  {
    "_id": ObjectId("6161e5b1a3eba3c7f2960d05"),
    "name": "Rectangle 3",
    "area": 12
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$let` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$limit`operatore in Amazon DocumentDB viene utilizzato per limitare il numero di documenti restituiti da una query. È simile all'operatore `$limit` MongoDB, ma ci sono alcune considerazioni specifiche quando lo si utilizza con Amazon DocumentDB.

In Amazon DocumentDB, l'`$limit`operatore è utile per l'impaginazione, in cui si desidera recuperare un sottoinsieme del totale dei documenti corrispondenti. Consente di controllare il numero di documenti restituiti in ogni risposta, migliorando le prestazioni e riducendo la quantità di dati trasferiti sulla rete.

Parametri

- `limit`: Il numero massimo di documenti da restituire. Deve essere un valore intero non negativo.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$limit`operatore per restituire al massimo un documento che corrisponde al filtro specificato.

Crea documenti di esempio

```
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" }
```

```
]);
```

Esempio di interrogazione

```
db.test.createIndex({ comments: "text" });

db.test.find({
  $and: [
    { star_rating: 5 },
    { $text: { $search: "couch" } }
  ]
}).limit(1);
```

Output

```
[ { _id: 2, star_rating: 5, comments: 'comfortable couch' } ]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$limit` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

elenco $ SearchIndexes

Nuovo dalla versione 8.0.1.

La fase di `$listSearchIndexes` aggregazione in Amazon DocumentDB restituisce informazioni sugli indici di ricerca esistenti in una raccolta. Deve essere la prima fase di una pipeline di aggregazione.

Parametri

- `name`: (opzionale) Il nome dell'indice di ricerca su cui restituire informazioni. Se omesso, vengono restituiti tutti gli indici di ricerca della raccolta.

Campi di output

Ogni documento restituito contiene i seguenti campi:

- `name`: il nome dell'indice di ricerca.
- `status`: lo stato di compilazione dell'indice. Uno tra `READY` (compilato e valido), `BUILDING` (è in corso una compilazione simultanea) o `FAILED` (la compilazione dell'indice non è stata completata correttamente).
- `queryable`: un valore booleano che indica se l'indice può essere attualmente utilizzato per fornire query. Questo è `false` per gli indici nascosti (che rimangonoREADY) e per gli indici. `FAILED` Un indice nascosto è il caso in cui `status` è ma è. `READY` `queryable` `false`
- `latestDefinitionVersion`: un documento contenente `version` (la versione del formato dell'indice) e `createdAt` (l'ora di creazione dell'indice).

Sintassi

```
db.collection.aggregate([
  { $listSearchIndexes: {} }
])

// Or filter by name:
db.collection.aggregate([
  { $listSearchIndexes: { name: "mySearchIndex" } }
])
```

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare lo `$listSearchIndexes` stage per elencare tutti gli indici di ricerca in una raccolta.

Esempio di interrogazione

```
db.movies.aggregate([
  { $listSearchIndexes: {} }
]);
```

Output

```
[
  {
    "name": "default",
    "status": "READY",
    "queryable": true,
    "latestDefinitionVersion": {
      "version": 2,
      "createdAt": ISODate("2026-05-11T21:12:57.974Z")
    }
  }
]
```

Un indice nascosto rimane `READY` ma riporta`queryable: false`, perché non può essere utilizzato per fornire interrogazioni mentre è nascosto:

```
[
  {
    "name": "myHiddenIndex",
    "status": "READY",
    "queryable": false,
```

```
    "latestDefinitionVersion": {
      "version": 2,
      "createdAt": ISODate("2026-05-11T21:12:57.974Z")
    }
  }
]
```

Per filtrare in base a un nome di indice specifico:

```
db.movies.aggregate([
  { $listSearchIndexes: { name: "default" } }
]);
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$listSearchIndexes` dello stage, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$literal`operatore in Amazon DocumentDB viene utilizzato per rappresentare un valore letterale all'interno di una fase di pipeline di aggregazione. Consente di includere un valore specifico, ad

esempio un numero, una stringa o un valore booleano, senza interpretarlo come riferimento di campo o espressione.

Questo operatore è particolarmente utile quando è necessario includere un valore letterale come parte di una pipeline di aggregazione più complessa, ad esempio quando si creano filtri di interrogazione dinamici o si eseguono calcoli.

Parametri

Nessuno

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'uso dell'`$literal`operatore per includere un valore letterale in una pipeline di aggregazione. L'`$literal`operatore viene utilizzato per includere il valore 18 come valore letterale nell'espressione $gt. Ciò consente alla pipeline di aggregazione di confrontare il campo di età con il valore letterale 18 per determinare se la persona è un adulto.

Crea documenti di esempio

```
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" }
]);
```

Esempio di query

```
db.collection.aggregate([
  {
    $project: {
      name: 1,
      age: 1,
      city: 1,
      isAdult: { $gt: ["$age", { $literal: 18 }] }
    }
  }
]);
```

Output

```
[
```

```
  {
    "_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
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$literal` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

Novità dalla versione 4.0.

L'`$ln`operatore in Amazon DocumentDB calcola il logaritmo naturale (base e) di un numero specificato. Restituisce il logaritmo del numero alla base e.

Parametri

- `expression`: Il numero per il quale verrà calcolato il logaritmo naturale.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$log`operatore per calcolare il logaritmo naturale di un numero.

Crea documenti di esempio

```
db.numbers.insertMany([
  { _id: 1, value: 10 },
  { _id: 2, value: 100 },
  { _id: 3, value: 1000 }
]);
```

Esempio di interrogazione

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    naturalLog: { $ln: "$value" }
  }}
]);
```

Output

```
[
  { "_id" : 1, "naturalLog" : 2.302585092994046 },
  { "_id" : 2, "naturalLog" : 4.605170185988092 },
  { "_id" : 3, "naturalLog" : 6.907755278982137 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$ln` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

Novità dalla versione 4.0.

L'`$log`operatore in Amazon DocumentDB calcola il logaritmo naturale di un numero. Restituisce il logaritmo in base e del numero specificato.

Parametri

- `expression`: Il numero per il quale verrà calcolato il logaritmo naturale.
- `base`: Valore base per calcolare il log.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$log`operatore per calcolare il logaritmo naturale di un numero.

Crea documenti di esempio

```
db.numbers.insertMany([
  { _id: 1, value: 10 },
  { _id: 2, value: 100 },
  { _id: 3, value: 1000 }
]);
```

Esempio di interrogazione

```
db.numbers.aggregate([
  { $project: {
```

```
    _id: 1,
    naturalLog: { $log: ["$value", 10] }
  }}
]);
```

Output

```
[
  { "_id" : 1, "naturalLog" : 1 },
  { "_id" : 2, "naturalLog" : 2 },
  { "_id" : 3, "naturalLog" : 2.9999999999999996 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$log` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

Novità dalla versione 4.0.

L'`$log10`operatore in Amazon DocumentDB viene utilizzato per calcolare il logaritmo in base 10 di un numero. È utile per eseguire calcoli logaritmici su campi numerici all'interno della pipeline di aggregazione.

Parametri

- `expression`: L'espressione numerica per la quale deve essere calcolato il logaritmo in base 10.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$log10`operatore per calcolare il logaritmo in base 10 di un campo numerico.

Crea documenti di esempio

```
db.numbers.insertMany([
  { _id: 1, value: 1 },
  { _id: 2, value: 10 },
  { _id: 3, value: 100 },
  { _id: 4, value: 1000 }
]);
```

Esempio di interrogazione

```
db.numbers.aggregate([
  {
    $project: {
      _id: 1,
      log10Value: { $log10: "$value" }
    }
  }
]);
```

Output

```
[
  { "_id": 1, "log10Value": 0 },
  { "_id": 2, "log10Value": 1 },
  { "_id": 3, "log10Value": 2 },
  { "_id": 4, "log10Value": 3 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$log10` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'operatore di `$lt` aggregazione confronta due valori e restituisce `true` se il primo valore è inferiore al secondo, altrimenti restituisce. `false`

Parametri

- `expression1`: il primo valore da confrontare.
- `expression2`: Il secondo valore da confrontare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$lt`operatore per identificare gli articoli in esaurimento.

Crea documenti di esempio

```
db.warehouse.insertMany([
  { _id: 1, item: "Bolts", stock: 5 },
  { _id: 2, item: "Nuts", stock: 25 },
  { _id: 3, item: "Screws", stock: 8 }
]);
```

Esempio di interrogazione

```
db.warehouse.aggregate([
  {
    $project: {
      item: 1,
      stock: 1,
      lowStock: { $lt: ["$stock", 10] }
    }
  }
]);
```

Output

```
[
  { _id: 1, item: 'Bolts', stock: 5, lowStock: true },
  { _id: 2, item: 'Nuts', stock: 25, lowStock: false },
  { _id: 3, item: 'Screws', stock: 8, lowStock: true }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di `$lt` aggregazione, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'operatore di `$lte` aggregazione confronta due valori e restituisce `true` se il primo valore è minore o uguale al secondo, altrimenti restituisce. `false`

Parametri

- `expression1`: il primo valore da confrontare.
- `expression2`: Il secondo valore da confrontare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$lte`operatore per identificare articoli economici.

Crea documenti di esempio

```
db.menu.insertMany([
  { _id: 1, dish: "Salad", price: 8 },
  { _id: 2, dish: "Pasta", price: 12 },
  { _id: 3, dish: "Soup", price: 6 }
]);
```

Esempio di interrogazione

```
db.menu.aggregate([
  {
    $project: {
      dish: 1,
      price: 1,
      affordable: { $lte: ["$price", 10] }
    }
  }
]);
```

Output

```
[
  { _id: 1, dish: 'Salad', price: 8, affordable: true },
  { _id: 2, dish: 'Pasta', price: 12, affordable: false },
  { _id: 3, dish: 'Soup', price: 6, affordable: true }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di `$lte` aggregazione, scegli la scheda relativa alla lingua che desideri utilizzare:

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

La fase `$lookup` di aggregazione in Amazon DocumentDB consente di eseguire un join esterno sinistro tra due raccolte. Questa operazione consente di combinare i dati di più raccolte in base ai valori dei campi corrispondenti. È particolarmente utile quando è necessario incorporare dati provenienti da raccolte correlate nei risultati delle query.

Parametri

- `from`: il nome della raccolta con cui eseguire il join.
- `localField`: Il campo dei documenti di input da confrontare con`foreignField`.
- `foreignField`: Il campo dei documenti della `from` raccolta da confrontare con il`localField`.
- `as`: il nome del nuovo campo da aggiungere ai documenti di output contenenti i documenti corrispondenti della `from` raccolta.
- `pipeline`: Facoltativo se `localField` e `foreignField` sono specificati. Una pipeline di aggregazione da eseguire sulla `from` raccolta. La pipeline restituisce i documenti della raccolta. `from` Per restituire tutti i documenti, specificate una pipeline `[]` vuota. La pipeline non può includere le fasi `$out` o`$merge`. Consulta Unisci condizioni e sottoquery correlate (MongoDB Shell).
- `let`: facoltativo. Un documento che definisce le variabili da utilizzare nelle `pipeline` fasi. Usa le espressioni variabili per accedere ai campi dei documenti di input che sono uniti a`pipeline`. Per fare riferimento a una variabile nelle fasi della pipeline, utilizzate la `"$$<variable>"` sintassi. Consulta Unisci condizioni e sottoquery correlate (MongoDB Shell).

Esempio (MongoDB Shell)

L'esempio seguente dimostra una semplice `$lookup` operazione che unisce i dati della `orders` raccolta alla raccolta. `customers`

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.customers.aggregate([
  {
    $lookup: {
      from: "orders",
      localField: "_id",
      foreignField: "customer_id",
      as: "orders"
    }
  }
]);
```

Output

```
[
  {
    _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 } ]
  }
]
```

Unisci condizioni e sottoquery correlate (MongoDB Shell)

Nuovo dalla versione 8.0.2.

Oltre a una singola corrispondenza di uguaglianza, `$lookup` in Amazon DocumentDB è possibile eseguire una pipeline di aggregazione sulla raccolta unita. Ciò supporta le sottoquery non correlate, che eseguono la stessa sottoquery per ogni documento di input, e le sottoquery correlate, che utilizzano variabili definite dai campi di ciascun documento di input per restituire risultati correlati a quel documento.

Sintassi

Per eseguire una sottoquery correlata, definisci le variabili dai campi del documento di input con e fai riferimento ad esse nelle fasi: `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>
  }
}
```

Parametri

- `let`: un documento che definisce le variabili dai campi dei documenti di input. Per fare riferimento a una variabile nelle fasi della pipeline, utilizzate la "`$$<variable>`" sintassi. È possibile accedere alle variabili dagli stadi in`pipeline`, inclusi gli `$lookup` stadi aggiuntivi annidati nel. `pipeline` Una `$match` fase richiede l'uso dell'`$expr`operatore per accedere alle variabili. Ometti `let` di eseguire una sottoquery non correlata.

- `pipeline`: la pipeline di aggregazione da eseguire sulla raccolta. `from` La pipeline non può accedere direttamente ai campi dei documenti di input; definisce invece le variabili con le variabili `let` e fa riferimento alle variabili. Per restituire tutti i documenti, specificate una pipeline `[]` vuota.

Gli operatori `$eq`$gt, `$gte`$lt, e di `$lte` confronto inseriti in un `$expr` operatore all'interno di uno `$match` stadio possono utilizzare un indice sulla `from` raccolta quando confrontano un campo della raccolta unita con una `let` variabile o una costante.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente unisce ogni cliente ai propri ordini e mantiene solo gli ordini il cui valore `total` è maggiore o uguale al `min_total` valore proprio di quel cliente. La condizione di join e quella per cliente filtrano entrambe le variabili di riferimento all'interno: `let` `$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"
    }
  }
]);
```

Output

```
[
  { _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: [] }
]
```

Per ogni cliente, la sottoquery viene valutata con i `min_total` valori del cliente `_id` e, pertanto, ogni documento riceve il proprio risultato correlato.

Sottoquery correlate utilizzando una sintassi concisa (MongoDB Shell)

Novità dalla versione 8.0.2.

Amazon DocumentDB supporta anche la sintassi concisa delle sottoquery correlate, che combina la corrispondenza di uguaglianza su e con un. `localField` `foreignField` `pipeline` La sintassi concisa elimina la necessità di esprimere la corrispondenza di uguaglianza all'interno di un operatore in una fase. `$expr` `$match`

Sintassi

```
{
  $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>
  }
}
```

Esempio di interrogazione

L'esempio seguente restituisce gli stessi risultati del precedente esempio di sottoquery correlata. La corrispondenza di uguaglianza tra `_id` e `customer_id` è espressa con `localField` e `foreignField` anziché con una `$expr` condizione:

```
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"
    }
  }
]);
```

Output

```
[
  { _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: [] }
]
```

Fasi supportate nelle sottoquery correlate

Quando viene utilizzata una `$lookup` pipeline`let`, le seguenti fasi possono fare riferimento alle variabili: `"$$<variable>"`

- `$match`(utilizzando l'`$expr`operatore)
- `$project`
- `$addFields`
- `$set`
- `$group`

- `$bucket`
- `$bucketAuto`
- `$sortByCount`
- `$lookup`(annidato)
- `$graphLookup`

Anche gli stadi che non valutano espressioni variabili`$sort`, come`$skip`,`$limit`,`$count`,`$sample`, e `$unwind``$unset`, possono essere utilizzati nella pipeline ma non possono fare riferimento alle variabili.

Le fasi seguenti non sono supportate in una `$lookup` pipeline che utilizzale`t`:`$facet`,`$redact`,`$replaceRoot`,,`$replaceWith`, `$geoNear` `$setWindowFields``$fill`, e `$match` with. `$text`

Restrizioni

- `pipeline`Non possono includere le `$merge` fasi `$out` o.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$lookup` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

Esempi di codice di sottoquery correlati

Per visualizzare un esempio di codice per l'utilizzo `$lookup` dello stage con una sottoquery correlata (utilizzando `let` and`pipeline`), scegli la scheda relativa alla lingua che desideri utilizzare:

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

Nuovo dalla versione 4.0.

Non supportato dal cluster Elastic.

L'`$ltrim`operatore in Amazon DocumentDB viene utilizzato per rimuovere i caratteri iniziali da una stringa. Per impostazione predefinita, rimuove gli spazi bianchi iniziali, ma puoi anche specificare un set di caratteri da rimuovere passando l'argomento chars.

Parametri

- `input`: La stringa di input da cui rimuovere i caratteri di spaziatura iniziali.
- `chars`: (opzionale) Per rimuovere caratteri specifici.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo di `$ltrim` per rimuovere i caratteri specificati (» *») dall'inizio di una stringa.

Crea documenti di esempio

```
db.collection.insertMany([
  { name: " *John Doe", age: 30 },
  { name: "Jane Doe*", age: 25 },
  { name: "  Bob Smith  ", age: 35 }
]);
```

Esempio di interrogazione

```
db.collection.aggregate([
  {
    $project: {
      _id: 0,
      name: {
        $ltrim: { input: "$name", chars: " *" }
      },
      age: 1
    }
  }
]);
```

Output

```
[
  { "name": "John Doe", "age": 30 },
  { "name": "Jane Doe ", "age": 25 },
  { "name": "Bob Smith  ", "age": 35 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$ltrim` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$map`operatore in Amazon DocumentDB consente di applicare un'espressione specificata a ciascun elemento di un array e restituire un nuovo array con gli elementi trasformati. Questo operatore è particolarmente utile per manipolare e trasformare i dati all'interno degli array, il che può contribuire a semplificare il codice dell'applicazione e migliorare le prestazioni delle query spingendo l'elaborazione dell'array a livello di database.

Parametri

- `input`: L'array da trasformare.
- `as`: (opzionale) Il nome della variabile da utilizzare all'interno dell'espressione in per rappresentare l'elemento corrente in fase di elaborazione.
- `in`: L'espressione da applicare a ogni elemento dell'array di input.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'operatore $map per trasformare una matrice di numeri, raddoppiando ogni valore.

Crea documenti di esempio

```
db.collection.insertMany([
  { _id: 1, numbers: [1, 2, 3, 4, 5] },
  { _id: 2, numbers: [10, 20, 30, 40, 50] }
])
```

Esempio di query

```
db.collection.aggregate([
  {
    $project: {
      doubledNumbers: { $map: { input: "$numbers", as: "num", in: { $multiply: ["$
$num", 2] } } }
    }
  }
])
```

Output

```
[
  { _id: 1, doubledNumbers: [2, 4, 6, 8, 10] },
  { _id: 2, doubledNumbers: [20, 40, 60, 80, 100] }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$map` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

La fase della `$match` pipeline in Amazon DocumentDB viene utilizzata per filtrare i documenti di input solo in base a quelli che corrispondono ai criteri di query specificati. È una delle fasi della pipeline più comunemente utilizzate nelle operazioni di aggregazione. La `$match` fase viene applicata prima di qualsiasi altra fase della pipeline, il che consente di ridurre in modo efficiente il numero di documenti che devono essere elaborati nelle fasi successive.

Parametri

- `query`: Un documento che esprime i criteri di selezione per l'operazione. Il documento di interrogazione utilizza la stessa sintassi del `find()` metodo.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'uso dello `$match` stage per filtrare i documenti in base a un valore di campo specifico.

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.collection.aggregate([
  { $match: { age: { $gt: 30 } } },
  { $project: { _id: 1, name: 1, city: 1 } }
]);
```

Output

```
[
  { "_id": 3, "name": "Bob", "city": "Chicago" },
  { "_id": 4, "name": "Alice", "city": "Miami" }
]
```

Lo `$match` stage filtra i documenti per includere solo quelli in cui il `age` campo è maggiore di 30.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$match` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'operatore `$max` di aggregazione restituisce il valore massimo di un campo specificato in un set di documenti. Questo operatore è utile per trovare il valore più alto in un set di documenti.

Parametri

- `expression`: L'espressione da utilizzare per calcolare il valore massimo.

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare l'`$max`operatore per trovare il punteggio massimo in una raccolta di documenti degli studenti. La `$group` fase raggruppa tutti i documenti e l'`$max`operatore viene utilizzato per calcolare il valore massimo del `score` campo in tutti i documenti.

Crea documenti di esempio

```
db.students.insertMany([
  { name: "John", score: 85 },
  { name: "Jane", score: 92 },
  { name: "Bob", score: 78 },
  { name: "Alice", score: 90 }
])
```

Esempio di interrogazione

```
db.students.aggregate([
  { $group: { _id: null, maxScore: { $max: "$score" } } },
  { $project: { _id: 0, maxScore: 1 } }
])
```

Output

```
[ { maxScore: 92 } ]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$max`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce il valore massimo per i documenti in ogni finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

> **Note**
>
> Se utilizzato come operatore di finestra in`$setWindowFields`, `$max` è limitato a 100 MB di dati intermedi. Un'operazione che supera questo limite restituisce un errore.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`ticker`, ordina ogni partizione in base `hour` e restituisce il prezzo più alto visto dall'inizio della partizione fino al documento corrente.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        highestSoFar: {
          $max: "$price",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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 }
```

```
]
```

Ogni documento viene ampliato con un numero massimo di caratteri `price` all'interno della partizione corrispondente`highestSoFar`, fino al documento corrente incluso.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$max` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano sia l'utilizzo dell'accumulatore (in`$group`) che l'utilizzo dell'operatore di finestra (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()
```

$max N

Nuovo dalla versione 8.0.1.

L'`$maxN`operatore in Amazon DocumentDB restituisce gli N valori maggiori. Se utilizzato come accumulatore in una `$group` fase, restituisce un array di N valori massimi in ciascun gruppo. Se utilizzato come operatore di espressione di matrice, restituisce il numero massimo di N elementi di un array.

Parametri

- `input`: Un'espressione che si risolve nella matrice o nel campo da cui restituire i valori massimin.
- n: Un'espressione che si risolve in un numero intero positivo che specifica il numero massimo di valori da restituire.

Comportamento

Comportamento dell'operatore di espressione di matrice

- Non è possibile specificare un valore n inferiore a`1`.
- `$maxN`filtra i `null` valori trovati nell'`input`array.
- Se il valore specificato n è maggiore o uguale al numero di elementi nell'`input`array, `$maxN` restituisce tutti gli elementi dell'`input`array.
- Se si `input` risolve in un valore non di matrice, l'operazione di aggregazione ha un errore.
- Se `input` contiene sia elementi numerici che stringhe, gli elementi stringa vengono ordinati prima degli elementi numerici in base all'ordine di confronto BSON.

Comportamento dell'accumulatore

- Se usato come accumulatore, n deve essere un'espressione integrale positiva che è costante o dipende dal `_id` valore di. `$group`
- `$maxN`filtra i valori nulli e mancanti.
- Se il gruppo contiene meno di n elementi, `$maxN` restituisce tutti gli elementi del gruppo.
- `$maxN`confronta i dati di input seguendo l'ordine di confronto BSON per determinare il tipo di output appropriato. Quando i dati di input contengono più tipi di dati, il tipo di `$maxN` output è il più alto nell'ordine di confronto.

Ordinamento degli output

`$maxN`restituisce i valori senza un ordinamento particolare. Se è necessario garantire un particolare ordinamento, utilizza `$topN` invece o racchiudi il risultato con. `$sortArray`

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$maxN`accumulatore per recuperare i due punteggi più alti per ogni materia.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.scores.aggregate([
  { $group: { _id: "$subject", highestTwo: { $maxN: { input: "$score", n: 2 } } } }
]);
```

Output

```
[
  { "_id": "math", "highestTwo": [85, 93] },
  { "_id": "science", "highestTwo": [88, 90] }
]
```

Esempio di utilizzo delle espressioni (MongoDB Shell)

L'`$maxN`operatore può essere utilizzato anche come espressione all'interno di uno `$project` stage per restituire gli N elementi più grandi da un campo di matrice.

Crea documenti di esempio

```
db.readings.insertMany([
  { _id: 1, sensor: "A", values: [45, 12, 78, 3, 56] },
  { _id: 2, sensor: "B", values: [90, 23, 67, 11, 44] }
]);
```

Esempio di interrogazione

```
db.readings.aggregate([
```

```
  { $project: {
      highestThree: { $maxN: { input: "$values", n: 3 } }
    }}
]);
```

Output

```
[
  { "_id": 1, "highestThree": [45, 56, 78] },
  { "_id": 2, "highestThree": [44, 90, 67] }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$maxN`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce una matrice con i valori n massimi per i documenti in ogni finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

> **Note**
>
> Se utilizzato come operatore di finestra in`$setWindowFields`, `$maxN` è limitato a 100 MB di dati intermedi. Un'operazione che supera questo limite restituisce un errore.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`ticker`, ordina ogni partizione in base `hour` e restituisce i due prezzi più alti visualizzati dall'inizio della partizione fino al documento corrente.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        highestTwoSoFar: {
          $maxN: { input: "$price", n: 2 },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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] }
]
```

Ogni documento è arricchito con `highestTwoSoFar` i due valori più grandi `price` all'interno della relativa finestra. Come indicato nella sezione Ordinamento dell'output, l'ordine dei valori all'interno dell'array restituito non è garantito.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo $`maxN` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano l'uso dell'accumulatore (in$`group`), l'uso delle espressioni (in$`project`) e l'uso dell'operatore di finestra (in$`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()
```

$mediana

Nuovo dalla versione 8.0.1.

L'`$median`operatore in Amazon DocumentDB calcola il valore mediano dei dati numerici. Come accumulatore, calcola la mediana dei valori numerici tra i documenti all'interno di un gruppo nella fase di una pipeline di aggregazione. `$group` Come espressione, calcola la mediana di una matrice di numeri.

Parametri

- `input`: Un'espressione che si risolve in un valore numerico o in una matrice di valori numerici.
- `method`: una stringa che specifica il metodo di calcolo. Attualmente `"approximate"` è supportato solo quello che utilizza l'algoritmo t-digest.

Comportamento

Il `"approximate"` metodo utilizza l'algoritmo t-digest per calcolare una mediana approssimativa. Il risultato è un valore esistente del set di dati anziché un'interpolazione tra valori. La precisione migliora all'aumentare del numero di punti dati.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$median`operatore per calcolare il punteggio mediano del test per classe.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.students.aggregate([
  { $group: {
      _id: "$class",
      medianScore: { $median: { input: "$score", method: "approximate" } }
    }}
]);
```

Output

```
[
  { "_id": "A", "medianScore": 85 },
  { "_id": "B", "medianScore": 80 }
]
```

Esempio di utilizzo delle espressioni (MongoDB Shell)

L'`$median`operatore può essere utilizzato anche come espressione all'interno di uno `$project` stage per calcolare la mediana di un campo di matrice.

Crea documenti di esempio

```
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] }
]);
```

Esempio di interrogazione

```
db.surveys.aggregate([
  { $project: {
      medianRating: { $median: { input: "$ratings", method: "approximate" } }
    }}
]);
```

Output

```
[
  { "_id": 1, "medianRating": 5 },
  { "_id": 2, "medianRating": 30 },
  { "_id": 3, "medianRating": 2 }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$median`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, calcola la mediana dei valori numerici per i documenti in ciascuna finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un documento. `window`

> **Note**
>
> Se utilizzato come operatore di finestra in`$setWindowFields`, `$median` è limitato a 100 MB di dati intermedi. Un'operazione che supera questo limite restituisce un errore.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base `city` e calcola la mediana `reading` tra tutti i documenti in ciascuna partizione.

```
db.temperatures.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$city",
      output: {
        medianReading: {
          $median: { input: "$reading", method: "approximate" },
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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 }
]
```

Ogni documento è aumentato con `medianReading` la mediana approssimativa di tutti i documenti nella relativa partizione. `reading`

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'`$median`operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano l'uso dell'accumulatore (in`$group`), l'uso delle espressioni (in`$project`) e l'uso dell'operatore di finestra (in`$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

L'operatore di `$meta` aggregazione accede ai metadati associati ai documenti in una pipeline di aggregazione. Viene comunemente utilizzato per recuperare i punteggi di ricerca testuali e ordinare i risultati in base alla pertinenza.

Parametri

- `textScore`: recupera il punteggio di ricerca testuale che indica la pertinenza del documento rispetto alla query di ricerca.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$meta`operatore in una pipeline di aggregazione per recuperare e ordinare i punteggi di ricerca testuali.

Crea documenti di esempio

```
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." }
]);
```

Esempio di interrogazione

```
db.articles.aggregate([
  { $match: { $text: { $search: "Python" } } },
  { $addFields: { score: { $meta: "textScore" } } },
  { $sort: { score: -1 } }
]);
```

Output

```
[
```

```
  {
    _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
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di `$meta` aggregazione, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

$unisci

Introdotto nella versione 8.0

La fase di `$merge` aggregazione in Amazon DocumentDB viene utilizzata per unire i risultati della precedente fase della pipeline in una raccolta mirata. Ciò è utile per aggiornare o inserire documenti in una raccolta di destinazione in base ai dati dei documenti di input.

La `$merge` fase consente di eseguire varie azioni in base alla condizione di corrispondenza tra i documenti di input e la raccolta di destinazione, ad esempio:

```
- Insert new documents
- Update existing documents
- Delete documents
- Fail the operation if there are any conflicts
```

Parametri

- `into`: (obbligatorio) Il nome della raccolta di destinazione in cui unire i documenti di input.

- on: (obbligatorio) I campi da utilizzare come condizione di corrispondenza tra i documenti di input e la raccolta di destinazione.
- `whenMatched`: (opzionale) L'azione da eseguire quando il documento di input corrisponde a un documento esistente nella raccolta di destinazione. I valori supportati sono: `"merge"`, `"replace"`, `"keepExisting"` e `"fail"`. Amazon DocumentDB non supporta la specificazione di una pipeline di aggregazione come valore. `whenMatched`
- `whenNotMatched`: (opzionale) L'azione da eseguire quando il documento di input non corrisponde a nessun documento nella raccolta di destinazione. I valori supportati sono: `"insert"` e`"fail"`.
- `bypassDocumentValidation`: (opzionale) Se impostato su`true`, i documenti scritti nella raccolta di destinazione dallo `$merge` stage ignorano qualsiasi regola di convalida dello schema (`$jsonSchema`) definita in quella raccolta. L'impostazione predefinita è`false`, che applica le regole di convalida della raccolta di destinazione. Questa funzionalità è supportata in Amazon DocumentDB 8.0.

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare la `$merge` fase per aggiornare una `users` raccolta con nuovi dati da una pipeline di input.

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.inputData.aggregate([
  {
    $merge: {
      into: "users",
      on: "_id",
```

```
      whenMatched: "merge",
      whenNotMatched: "insert"
    }
  }
])
```

Output

Dopo aver eseguito la `$merge` pipeline, la `users` raccolta conterrà i seguenti documenti:

```
[
  { _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" }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$merge` comando, scegliete la scheda corrispondente alla lingua che desiderate utilizzare:

Node.js

Ecco un esempio di utilizzo dell'operatore $merge in un' Node.js applicazione:

```
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

Ecco un esempio di utilizzo dell'operatore $merge in un'applicazione 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']

    # 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

L'`$mergeObjects`operatore in Amazon DocumentDB viene utilizzato per combinare più documenti o oggetti in un unico documento. Questo operatore è particolarmente utile quando è necessario unire il contenuto di due o più documenti o oggetti, sovrascrivendo potenzialmente i valori di un oggetto con quelli di un altro.

Parametri

- `expression1`: Il primo oggetto da unire.
- `expression2`: (opzionale) Il secondo oggetto da unire.
- `expression3`: (opzionale) Oggetti aggiuntivi da unire.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$mergeObjects`operatore per combinare due oggetti.

Crea documenti di esempio

```
db.collection.insertMany([
  { "_id": 1, "name": "John", "address": { "city": "New York", "state": "NY" } },
  { "_id": 2, "name": "Jane", "address": { "city": "Los Angeles", "state": "CA" } }
]);
```

Esempio di query

```
db.collection.aggregate([
  {
    $project: {
      "combinedAddress": {
        $mergeObjects: ["$address", { "country": "USA" }]
      }
    }
  }
])
```

Output

```
[
  {
    "_id": 1,
```

```
    "combinedAddress": {
      "city": "New York",
      "state": "NY",
      "country": "USA"
    }
  },
  {
    "_id": 2,
    "combinedAddress": {
      "city": "Los Angeles",
      "state": "CA",
      "country": "USA"
    }
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$mergeObjects` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$millisecond`operatore in Amazon DocumentDB viene utilizzato per estrarre la parte in millisecondi di un valore di data.

Parametri

Nessuno

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'`$millisecond`operatore per estrarre la parte in millisecondi di un valore di data.

Crea documenti di esempio

```
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")
  }
])
```

Esempio di interrogazione

```
db.events.aggregate([
  {
    $project: {
      name: 1,
      milliseconds: { $millisecond: "$timestamp" }
    }
  }
])
```

Output

```
[
  {
    "_id": ObjectId("644332a42054ed1b0d15f0c1"),
    "name": "Event 1",
    "milliseconds": 123
  },
  {
```

```
    "_id": ObjectId("644332a42054ed1b0d15f0c2"),
    "name": "Event 2",
    "milliseconds": 456
  },
  {
    "_id": ObjectId("644332a42054ed1b0d15f0c3"),
    "name": "Event 3",
    "milliseconds": 789
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$millisecond` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'operatore `$min` di aggregazione restituisce il valore minimo di un campo specificato in un set di documenti. Questo operatore è utile per trovare il valore più basso in un set di documenti.

Parametri

- `expression`: L'espressione da valutare. Può essere un percorso di campo, una variabile o qualsiasi espressione che si risolve in un valore.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$min`operatore per trovare il valore minimo del `age` campo in più documenti.

Crea documenti di esempio

```
db.users.insertMany([
```

```
  { name: "John", age: 35 },
  { name: "Jane", age: 28 },
  { name: "Bob", age: 42 },
  { name: "Alice", age: 31 }
]);
```

Esempio di interrogazione

```
db.users.aggregate([
  { $group: { _id: null, minAge: { $min: "$age" } } },
  { $project: { _id: 0, minAge: 1 } }
])
```

Output

```
[ { minAge: 28 } ]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$min`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce il valore minimo per i documenti in ogni finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

> **Note**
>
> Se utilizzato come operatore di finestra in`$setWindowFields`, `$min` è limitato a 100 MB di dati intermedi. Un'operazione che supera questo limite restituisce un errore.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`ticker`, ordina ogni partizione in base `hour` e restituisce il prezzo più basso visto dall'inizio della partizione fino al documento corrente.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        lowestSoFar: {
          $min: "$price",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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 }
]
```

Ogni documento è integrato`lowestSoFar`, per un minimo `price` all'interno della sua partizione, fino al documento corrente incluso.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$min` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano sia l'utilizzo dell'accumulatore (in`$group`) che l'utilizzo dell'operatore di finestra (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()
```

$minN

Nuovo dalla versione 8.0.1.

L'`$minN`operatore in Amazon DocumentDB restituisce gli N valori più piccoli. Se utilizzato come accumulatore in una `$group` fase, restituisce un array di N valori minimi in ciascun gruppo. Se utilizzato come operatore di espressione di matrice, restituisce gli N elementi minimi di un array.

Parametri

- `input`: Un'espressione che si risolve nella matrice o nel campo da cui restituire i valori minimin.
- n: Un'espressione che si risolve in un numero intero positivo che specifica quanti valori minimi restituire.

Comportamento

Comportamento dell'operatore di espressione di matrice

- Non è possibile specificare un valore n inferiore a `1`.
- `$minN`filtra i `null` valori trovati nell'`input`array.
- Se il valore specificato n è maggiore o uguale al numero di elementi nell'`input`array, `$minN` restituisce tutti gli elementi dell'`input`array.
- Se si `input` risolve in un valore non di matrice, l'operazione di aggregazione ha un errore.
- Se `input` contiene sia elementi numerici che stringhe, gli elementi numerici vengono ordinati prima degli elementi stringa in base all'ordine di confronto BSON.

Comportamento dell'accumulatore

- Se usato come accumulatore, n deve essere un'espressione integrale positiva che è costante o dipende dal `_id` valore di. `$group`
- `$minN`filtra i valori nulli e mancanti.
- Se il gruppo contiene meno di n elementi, `$minN` restituisce tutti gli elementi del gruppo.
- `$minN`confronta i dati di input seguendo l'ordine di confronto BSON per determinare il tipo di output appropriato. Quando i dati di input contengono più tipi di dati, il tipo di `$minN` output è il più basso nell'ordine di confronto.

Ordinamento degli output

`$minN`restituisce i valori senza un ordinamento particolare. Se è necessario garantire un particolare ordinamento, utilizza `$bottomN` invece o racchiudi il risultato con. `$sortArray`

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$minN`accumulatore per recuperare i due punteggi più bassi per ogni materia.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.scores.aggregate([
  { $group: { _id: "$subject", lowestTwo: { $minN: { input: "$score", n: 2 } } } }
]);
```

Output

```
[
  { "_id": "math", "lowestTwo": [72, 68] },
  { "_id": "science", "lowestTwo": [78, 65] }
]
```

Esempio di utilizzo delle espressioni (MongoDB Shell)

L'`$minN`operatore può essere utilizzato anche come espressione all'interno di uno `$project` stage per restituire gli N elementi più piccoli da un campo di matrice.

Crea documenti di esempio

```
db.readings.insertMany([
  { _id: 1, sensor: "A", values: [45, 12, 78, 3, 56] },
  { _id: 2, sensor: "B", values: [90, 23, 67, 11, 44] }
]);
```

Esempio di interrogazione

```
db.readings.aggregate([
  { $project: {
      lowestThree: { $minN: { input: "$values", n: 3 } }
    }}
]);
```

Output

```
[
  { "_id": 1, "lowestThree": [45, 12, 3] },
  { "_id": 2, "lowestThree": [44, 23, 11] }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$minN`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce una matrice contenente fino ai valori n più piccoli per i documenti in ogni finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

> **Note**
>
> Se utilizzato come operatore di finestra in`$setWindowFields`, `$minN` è limitato a 100 MB di dati intermedi. Un'operazione che supera questo limite restituisce un errore.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`ticker`, ordina ogni partizione in base `hour` e restituisce i due prezzi più bassi visualizzati dall'inizio della partizione fino al documento corrente.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
```

```
      sortBy: { hour: 1 },
      output: {
        lowestTwoSoFar: {
          $minN: { input: "$price", n: 2 },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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] }
]
```

Ogni documento è arricchito con `lowestTwoSoFar` i due valori più piccoli `price` all'interno della relativa finestra. Come indicato nella sezione Ordinamento dell'output, l'ordine dei valori all'interno dell'array restituito non è garantito.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo $`minN` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano l'uso dell'accumulatore (in$`group`), l'uso delle espressioni (in$`project`) e l'uso dell'operatore di finestra (in$`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

La fase della pipeline di `$minute` aggregazione in Amazon DocumentDB estrae il valore minuto da un campo di data o timestamp.

Questo operatore è utile quando è necessario eseguire calcoli o raggruppamenti basati su data e ora all'interno della pipeline di aggregazione.

Parametri

- `expression`: Il campo della data o dell'ora da cui estrarre il valore dei minuti.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$minute`operatore per raggruppare i documenti in base al valore minuto estratto dal campo timestamp e contare il numero di documenti in ogni gruppo.

Crea documenti di esempio

```
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") }
]);
```

Esempio di interrogazione

```
db.events.aggregate([
```

```
  {
    $group: {
      _id: {
        minute: { $minute: "$timestamp" }
      },
      count: { $count: {} }
    }
  },
  { $sort: { "_id.minute": 1 } }
]);
```

Output

```
[
  { "_id": { "minute": 30 }, "count": 2 },
  { "_id": { "minute": 31 }, "count": 2 },
  { "_id": { "minute": 32 }, "count": 1 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$minute` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$mod`operatore è un operatore aritmetico che esegue un'operazione modulo su un numero. Restituisce il resto della divisione di un numero per un altro. Questo operatore viene comunemente utilizzato per determinare se un numero è pari o dispari o per distribuire gli elementi in un numero finito di gruppi.

Parametri

- `expression1`: L'espressione del dividendo.
- `expression2`: L'espressione del divisore.

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'`$mod`operatore per determinare il numero di widget rimanenti durante la spedizione in pacchi da 100.

Crea documenti di esempio

```
db.widgets.insertMany([
  { "_id" : 1, "widget" : "A", "count" : 80372 },
  { "_id" : 2, "widget" : "B", "count" : 409282 },
  { "_id" : 3, "widget" : "C", "count" : 60739 }
])
```

Esempio di interrogazione

```
db.widgets.aggregate([
```

```
  { $addFields: { leftOver: { $mod: [ "$count", 100 ] } } }
])
```

Output

```
[
  { "_id" : 1, "widget" : "A", "count" : 80372, "leftOver" : 72 },
  { "_id" : 2, "widget" : "B", "count" : 409282, "leftOver" : 82 },
  { "_id" : 3, "widget" : "C", "count" : 60739, "leftOver" : 39 }
]
```

L'output mostra il resto del valore `count` diviso per 100 per ogni documento, che rappresenta il numero di widget rimanenti durante la spedizione in pacchi da 100.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del $mod comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$month`operatore in Amazon DocumentDB restituisce il mese di una data come numero compreso tra 1 e 12. Questo operatore è utile per estrarre il componente del mese da un campo di data ed eseguire aggregazioni e analisi basate sulle date.

Parametri

- `date_expression`: Questa è l'espressione o il campo che contiene la data o il timestamp da cui si desidera estrarre il mese.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$month`operatore per estrarre il mese da un campo data e raggruppare i dati per mese.

Crea documenti di esempio

```
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") }
]);
```

Esempio di interrogazione

```
db.sales.aggregate([
  { $group: {
      _id: { month: { $month: "$date" } },
      totalSales: { $sum: "$price" }
    }},
  { $sort: { "_id.month": 1 } }
]);
```

Output

```
[
  { _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 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$month` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$multiply`operatore in Amazon DocumentDB viene utilizzato per moltiplicare i valori di due o più campi o espressioni. Questo operatore è particolarmente utile per eseguire operazioni aritmetiche su campi numerici all'interno di documenti. Può essere utilizzato in varie fasi della pipeline di aggregazione, ad esempio `$project` e`$addFields`, per creare nuovi campi o modificare quelli esistenti.

Parametri

- `expression1`: La prima espressione numerica da moltiplicare.
- `expression2`: La seconda espressione numerica da moltiplicare.
- `[expression3, ...]`: (opzionale) Espressioni numeriche aggiuntive da moltiplicare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo di `$multiply` to calculing `bonus_miles` moltiplicando `base_miles` e `bonus_rate` per i clienti che hanno utilizzato una carta di credito per il viaggio.

Documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
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"]
      }
    }
  }
]);
```

Output

```
[
  { _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 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$multiply` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$naturale

L'`$natural`operatore in Amazon DocumentDB viene utilizzato per ordinare i documenti nel loro ordine naturale, ovvero l'ordine in cui sono stati inseriti nella raccolta. Ciò è in contrasto con il

comportamento di ordinamento predefinito, che consiste nell'ordinare i documenti in base ai valori dei campi specificati.

Parametri

Nessuno

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$natural`operatore per ordinare i documenti in una raccolta nel loro ordine naturale.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.people.find({}, { "_id": 1, "name": 1 }).sort({ "$natural": 1 });
```

Output

```
[
  { "_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" }
]
```

La query ordina i documenti della raccolta nel loro ordine naturale, che è l'ordine in cui sono stati inseriti.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$natural` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'operatore di `$ne` aggregazione confronta due valori e restituisce `true` se non sono uguali, altrimenti restituisce. `false`

Parametri

- `expression1`: Il primo valore da confrontare.
- `expression2`: Il secondo valore da confrontare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$ne`operatore per identificare gli ordini con modifiche di stato.

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.orders.aggregate([
  {
    $project: {
      orderId: 1,
      status: 1,
      expectedStatus: 1,
      needsAttention: { $ne: ["$status", "$expectedStatus"] }
    }
  }
]);
```

Output

```
[
  { _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 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di $ne aggregazione, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'operatore `$not` di aggregazione esegue un'operazione NOT logica su un'espressione. Restituisce `true` se l'espressione restituisce a `false` e `false` se l'espressione restituisce a. `true`

Parametri

- `expression`: L'espressione da negare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$not`operatore per invertire i valori booleani.

Crea documenti di esempio

```
db.users.insertMany([
  { _id: 1, name: "Alice", active: true },
  { _id: 2, name: "Bob", active: false },
  { _id: 3, name: "Charlie", active: true }
```

```
]);
```

Esempio di interrogazione

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      active: 1,
      inactive: { $not: ["$active"] }
    }
  }
]);
```

Output

```
[
  { _id: 1, name: 'Alice', active: true, inactive: false },
  { _id: 2, name: 'Bob', active: false, inactive: true },
  { _id: 3, name: 'Charlie', active: true, inactive: false }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di `$not` aggregazione, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

$oggetto ToArray

L'operatore di `$objectToArray` aggregazione in Amazon DocumentDB converte un oggetto (o documento) in un array. L'input per l'operatore è un documento e l'output è costituito da un elemento

di matrice per ogni coppia campo-valore nel documento di input. Questo operatore è utile quando è necessario utilizzare i singoli campi di un documento come matrice, ad esempio quando si desidera trovare il documento con il valore massimo o minimo per un determinato campo.

Parametri

- `expression`: L'espressione del documento da convertire in una matrice.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$objectToArray`operatore per trovare il documento con l'inventario massimo per una catena di negozi di noleggio video.

Crea documenti di esempio

```
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
    }
  }
]);
```

Esempio di interrogazione

```
db.videos.aggregate([
  {
    $project: {
      name: 1,
      videos: {
        $objectToArray: "$inventory"
      }
    }
  },
  {
    $unwind: "$videos"
  },
  {
    $group: {
      _id: "$name",
      maxInventory: {
        $max: "$videos.v"
      }
    }
  }
]);
```

Output

```
[
  {
    "_id": "Bravemind",
    "maxInventory": 1000
  },
  {
    "_id": "Live Soft",
    "maxInventory": 1000
  },
```

```
  {
    "_id": "Romancing the Rock",
    "maxInventory": 500
  },
  {
    "_id": "Top Pilot",
    "maxInventory": 1000
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$objectToArray` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'operatore di `$or` aggregazione valuta più espressioni e restituisce `true` se almeno un'espressione restituisce lo stesso risultato. `true` Restituisce `false` solo se tutte le espressioni lo sono. `false`

Parametri

- `expressions`: Una serie di espressioni da valutare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo `$or` dell'operatore per verificare se i prodotti soddisfano uno qualsiasi dei criteri multipli.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.items.aggregate([
  {
    $project: {
      name: 1,
      price: 1,
      onSale: 1,
      goodDeal: {
        $or: [
          { $lt: ["$price", 50] },
          { $eq: ["$onSale", true] }
```

```
        ]
      }
    }
  }
]);
```

Output

```
[
  { _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 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di `$or` aggregazione, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$out`operatore in Amazon DocumentDB viene utilizzato per scrivere il risultato di una pipeline di aggregazione in una raccolta specificata.

`$out`dovrebbe essere l'ultima fase della pipeline.

Parametri

- `output_collection`: il nome della raccolta di output in cui scrivere i risultati dell'aggregazione.

Nota: se la raccolta esiste già, verrà sostituita con i risultati della fase di aggregazione.

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare l'`$out`operatore in Amazon DocumentDB per scrivere i risultati di una pipeline di aggregazione in una nuova raccolta.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.products.aggregate([
  { $group: { _id: "$category", totalPrice: { $sum: "$price" } } },
  { $out: "product_categories" }
])
```

Output

Nessuno (i risultati vengono scritti nella raccolta di output).

La pipeline di aggregazione raggruppa i prodotti per categoria e calcola il prezzo totale degli articoli per ciascuna categoria. L'`$out`operatore scrive i risultati in una nuova raccolta denominata «product_categories».

Per visualizzare i risultati nella raccolta di output:

```
db.product_categories.find()
[
{ "_id" : "Books", "totalPrice" : 125 },
{ "_id" : "Electronics", "totalPrice" : 300 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$out` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$percentile

Nuovo dalla versione 8.0.1.

L'`$percentile`operatore in Amazon DocumentDB calcola i valori percentili specificati per i dati numerici. Come accumulatore, calcola i valori percentili tra i documenti all'interno di un gruppo nella fase di una pipeline di aggregazione. `$group` Come espressione, calcola i percentili per una serie di numeri.

Parametri

- `input`: Un'espressione che si risolve in un valore numerico o in una matrice di valori numerici.
- p: una matrice di valori percentili compresi tra 0 e 1, in cui ogni valore rappresenta un percentile da calcolare. Ad esempio, `[0.25, 0.5, 0.75]` calcola il 25°, 50° e 75° percentile.

- `method`: una stringa che specifica il metodo di calcolo. Attualmento solo `"approximate"` è supportato.

Comportamento

Il `"approximate"` metodo utilizza l'algoritmo t-digest per calcolare metriche approssimative basate sui percentili. Con set di dati di piccole dimensioni, valori percentili distinti possono portare allo stesso risultato. Ad esempio, con solo 5 valori per gruppo, p90 e p99 possono entrambi restituire il valore massimo del gruppo. La precisione migliora all'aumentare del numero di punti dati.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$percentile`operatore per calcolare il 25° e il 75° percentile dei punteggi per classe.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.students.aggregate([
  { $group: {
      _id: "$class",
      percentiles: { $percentile: { input: "$score", p: [0.25, 0.75], method:
 "approximate" } }
    }}
]);
```

Output

```
[
  { "_id": "A", "percentiles": [72, 90] },
  { "_id": "B", "percentiles": [75, 88] }
]
```

Esempio di utilizzo delle espressioni (MongoDB Shell)

L'`$percentile`operatore può essere utilizzato anche come espressione all'interno di uno `$project` stage per calcolare i percentili di un campo di matrice.

Crea documenti di esempio

```
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] }
]);
```

Esempio di interrogazione

```
db.surveys.aggregate([
  { $project: {
      quartiles: { $percentile: { input: "$responses", p: [0.25, 0.5, 0.75], method:
 "approximate" } }
    }}
]);
```

Output

```
[
  { "_id": 1, "quartiles": [6, 10, 16] },
  { "_id": 2, "quartiles": [5, 9, 15] }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$percentile`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, calcola i valori percentili specificati per i documenti in ciascuna finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un documento. `window`

> **Note**
>
> Se utilizzato come operatore di finestra in`$setWindowFields`, `$percentile` è limitato a 100 MB di dati intermedi. Un'operazione che supera questo limite restituisce un errore.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base `sensor` e calcola il 25° e il 75° percentile `reading` tra tutti i documenti di ciascuna partizione.

```
db.sensorData.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$sensor",
      output: {
        readingPercentiles: {
          $percentile: { input: "$reading", p: [0.25, 0.75], method: "approximate" },
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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] }
]
```

Ogni documento è aumentato con `readingPercentiles` i percentili approssimativi del 25° e 75° percentile di tutti i documenti nella sua partizione. `reading`

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'`$percentile`operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano l'uso dell'accumulatore (in`$group`), l'uso delle espressioni (in`$project`) e l'uso dell'operatore di finestra (in`$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

Introdotto nella versione 8.0

L'$powoperatore in Amazon DocumentDB consente di elevare un numero a una potenza. Questo può essere utile per eseguire calcoli esponenziali all'interno della pipeline di aggregazione.

Parametri

- <number>(richiesto): Il numero da elevare a potenza.
- <exponent>(richiesto): La potenza a cui deve essere elevato il numero.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'$powoperatore per calcolare il quadrato di un numero.

Crea documenti di esempio

```
db.numbers.insertMany([
```

```
  { "_id": 1, "value": 2 },
  { "_id": 2, "value": 3 },
  { "_id": 3, "value": 4 }
]);
```

Esempio di interrogazione

```
db.numbers.aggregate([
  { $addFields: { "square": { $pow: ["$value", 2] } } }
])
```

Output

```
[
  { "_id": 1, "value": 2, "square": 4 },
  { "_id": 2, "value": 3, "square": 9 },
  { "_id": 3, "value": 4, "square": 16 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del $pow comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

Node.js

Ecco un esempio di utilizzo dell'operatore $pow in un' Node.js applicazione:

```
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

Ecco un esempio di utilizzo dell'operatore $pow in un'applicazione 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([
        { "$addFields": { "square": { "$pow": ["$value", 2] } } }
    ]))

    print(result)
    client.close()

example()
```

$push

L'operatore `$push` di aggregazione restituisce un array di tutti i valori di un'espressione specificata per ogni gruppo. In genere viene utilizzato all'interno dello `$group` stage per accumulare valori in una matrice.

Parametri

- `expression`: L'espressione da valutare per ogni documento del gruppo.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$push`operatore per raccogliere tutti i nomi di prodotto per ciascuna categoria.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.sales.aggregate([
  {
    $group: {
      _id: "$category",
      products: { $push: "$product" }
    }
  }
]);
```

Output

```
[
  { _id: 'Furniture', products: [ 'Desk', 'Chair' ] },
  { _id: 'Electronics', products: [ 'Laptop', 'Mouse', 'Keyboard' ] }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$push`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce una matrice di tutti i valori dell'espressione specificata per i documenti in ogni finestra, in ordine di ordinamento. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

> **Note**
>
> Se utilizzato come operatore di finestra in`$setWindowFields`, `$push` è limitato a 100 MB di dati intermedi. Un'operazione che supera questo limite restituisce un errore.

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`category`, ordina ogni partizione in base `day` e restituisce un elenco continuo di `product` valori dall'inizio della partizione fino al documento corrente.

```
db.purchases.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$category",
      sortBy: { day: 1 },
      output: {
        productsSoFar: {
          $push: "$product",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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" ] }
]
```

Ogni documento è arricchito con `productsSoFar` l'array di tutti i `product` valori all'interno della partizione dall'inizio al documento corrente, in ordine di ordinamento.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$push` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano sia l'utilizzo dell'accumulatore (in`$group`) che l'utilizzo dell'operatore di finestra (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

L'`$project`operatore in Amazon DocumentDB consente di includere o escludere selettivamente campi dai documenti di output, passare valori alla fase successiva della pipeline e calcolare nuovi campi dai valori dei documenti di input.

Parametri

- `field`: Il campo da includere o escludere dai documenti di output può essere un percorso di campo (ad esempio, «a.b.c»).
- `1`oppure`true`: include il campo nell'output.
- `0`oppure`false`: esclude il campo dall'output.

Esempio (MongoDB Shell)

L'esempio seguente mostra l'utilizzo dell'`$project`operatore nella raccolta degli studenti

Crea documenti di esempio

```
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" }
  ]);
```

Questa query include solo i `math` campi `name` e nell'output. Il `_id` campo è incluso per impostazione predefinita a meno che non sia esplicitamente escluso.

```
db.students.aggregate([
  { $project: { "name": 1, "math": 1 } }
])
```

Output

```
{ _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 }
```

Questa query esclude i `_id` campi `grade` and dall'output, mostrando tutti gli altri campi (`name`,`math`,`science`).

```
db.students.aggregate([
  { $project: { "grade": 0, "_id": 0 } }
])
```

Output

```
{ 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 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$project` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$radianti ToDegrees

Nuovo dalla versione 8.0.1.

L'`$radiansToDegrees`operatore in Amazon DocumentDB converte un valore da radianti a gradi.

Parametri

- `expression`: Un'espressione che si risolve in un valore numerico in radianti.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$radiansToDegrees`operatore per convertire i valori radianti in gradi.

Create documenti di esempio

```
db.angles.insertMany([
  { "_id": 1, "radians": 0 },
```

```
  { "_id": 2, "radians": 1.5707963267948966 },
  { "_id": 3, "radians": 3.141592653589793 },
  { "_id": 4, "radians": 6.283185307179586 }
]);
```

Esempio di interrogazione

```
db.angles.aggregate([
  { $project: {
    "degrees": { $radiansToDegrees: "$radians" }
  }}
]);
```

Output

```
[
  { "_id": 1, "degrees": 0 },
  { "_id": 2, "degrees": 90 },
  { "_id": 3, "degrees": 180 },
  { "_id": 4, "degrees": 360 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$radiansToDegrees` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

Nuovo dalla versione 8.0

L'`$rand`operatore in Amazon DocumentDB viene utilizzato per generare un numero casuale compreso tra 0 e 1.

Parametri

Nessuno

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$rand`operatore per selezionare casualmente due documenti dalla raccolta. `temp`

Crea documenti di esempio

```
db.items.insertMany([
  { "name": "pencil", "quantity": 110 },
  { "name": "pen", "quantity": 159 }
])
```

Esempio di query

```
db.items.aggregate([
  {
    $project: {
      randomValue: { $rand: {} }
    }
  }
])
```

Output

```
[
  {
    _id: ObjectId('6924a5edd66dcae121d29517'),
    randomValue: 0.8615243955294392
  },
  {
    _id: ObjectId('6924a5edd66dcae121d29518'),
    randomValue: 0.22815483022099903
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$rand` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'operatore di `$range` aggregazione in Amazon DocumentDB viene utilizzato per creare una matrice di numeri consecutivi all'interno di un intervallo specificato. Questo operatore è particolarmente utile per generare sequenze di numeri, ad esempio gli indicatori di miglia per i punti di ristoro durante una gara, come illustrato negli esempi seguenti.

Parametri

- `start`: Il valore iniziale dell'intervallo.
- `end`: Il valore finale dell'intervallo.
- `step`: (opzionale) Il valore del passo da utilizzare per generare l'intervallo. Se non viene fornito, il valore predefinito del passo è 1.

Esempio (MongoDB Shell)

In questo esempio, utilizzeremo l'`$range`operatore per generare gli indicatori di miglio per le stazioni idriche in una gara ciclistica.

Crea documenti di esempio

```
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 }
]);
```

Esempio di query

```
db.races.aggregate([
```

```
  {
    $project: {
      race: 1,
      "waterStations": { $range: [20, "$distance", 20] }
    }
  }
]);
```

Output

```
[
  {
    _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 ] }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$range` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$rango

Nuovo dalla versione 8.0.2.

L'operatore `$rank` window in Amazon DocumentDB restituisce la posizione di ogni documento all'interno della sua partizione, in base all'ordine delle `$setWindowFields` fasi. `sortBy` I documenti con lo stesso `sortBy` valore ricevono la stessa classificazione e il valore distinto successivo salta le posizioni intermedie, lasciando spazi vuoti (ad esempio). `1, 2, 2, 4` È specificato `output` nel campo dello stage. `$setWindowFields`

È un operatore di finestra che viene utilizzato solo nello `$setWindowFields` stage.

Parametri

- `$rank`accetta `{}` come valore un documento vuoto.
- Non accetta una `window` cornice.
- Richiede esattamente una `sortBy` chiave sul `$setWindowFields` palco.

Esempio (MongoDB Shell)

L'esempio seguente suddivide i documenti in base`game`, ordina ogni partizione in ordine decrescente e `score` assegna a ciascun giocatore un grado.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.players.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$game",
      sortBy: { score: -1 },
      output: {
        rank: { $rank: {} }
      }
    }
  }
]);
```

Output

```
[
  { "_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 }
]
```

All'interno di ogni `game` partizione, i giocatori sono classificati in ordine decrescente`score`. Bob e Carol sono pari `85` e condividono il grado`2`, quindi il grado successivo passa a Dave. `4`

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$rank` dell'operatore nella `$setWindowFields` fase, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

La fase di `$redact` aggregazione in Amazon DocumentDB viene utilizzata per includere o escludere selettivamente contenuti dai documenti di output in base ai valori dei campi specificati. Ciò è particolarmente utile in scenari in cui è necessario controllare la visibilità dei dati sensibili in base ai livelli di accesso o alle autorizzazioni degli utenti.

Parametri

- `$cond`: espressione che restituisce uno `$$KEEP` o `$$DESCEND` più campi del documento. `$$PRUNE`
- `$$KEEP`: mantiene il campo corrente nel documento di output.
- `$$PRUNE`: rimuove il campo corrente dal documento di output.
- `$$DESCEND`: applica in modo ricorsivo lo `$redact` stage al campo corrente, che è un oggetto o un array.

Esempio (MongoDB Shell)

In questo esempio, useremo lo `$redact` stage per filtrare gli ordini in base al loro stato, mostrando solo gli ordini con valori di stato specifici.

Crea documenti di esempio

```
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" }
])
```

Esempio di query

```
db.orders.aggregate([
  {
    $redact: {
      $cond: {
        if: { $in: ["$status", ["shipped", "processing"]] },
        then: "$$KEEP",
        else: "$$PRUNE"
      }
    }
  }
])
```

Output

```
[
  { _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 questo esempio, lo `$redact` stage controlla il valore del `status` campo in ogni documento. Se `status` è «spedito» o «in elaborazione», il documento viene conservato ($$KEEP). Altrimenti, il documento viene eliminato ()$$PRUNE.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$redact` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'operatore di `$reduce` aggregazione in Amazon DocumentDB viene utilizzato per applicare cumulativamente una funzione di due argomenti agli elementi di un array per ridurre l'array a un singolo valore. Questo operatore è particolarmente utile per eseguire calcoli o trasformazioni complessi sui dati dell'array all'interno della pipeline di aggregazione.

Parametri

- `input`: L'array da ridurre.
- `initialValue`: Il valore iniziale da utilizzare nell'operazione di riduzione.
- `in`: L'espressione da valutare su ogni elemento dell'`input`array. Questa espressione dovrebbe restituire un valore che verrà utilizzato nella successiva iterazione della riduzione.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$reduce`operatore per calcolare la somma di tutti gli elementi di un array.

Crea documenti di esempio

```
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] }
])
```

Esempio di interrogazione

```
db.orders.aggregate([
  {
    $project: {
      total: {
        $reduce: {
          input: "$items",
          initialValue: 0,
```

```
          in: { $add: ["$$value", "$$this"] }
        }
      }
    }
  }
])
```

Output

```
[
  { "_id": 1, "total": 15 },
  { "_id": 2, "total": 60 },
  { "_id": 3, "total": 80 },
  { "_id": 4, "total": 300 }
]
```

L'`$reduce`operatore esegue un'iterazione sull'`items`array, aggiungendo ogni elemento al valore `initialValue` di 0. Il risultato è la somma di tutti gli elementi dell'array.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$reduce` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

Node.js

Ecco un esempio di utilizzo dell'`$reduce`operatore in un' Node.js applicazione:

```
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

Ecco un esempio di utilizzo dell'`$reduce`operatore in un'applicazione 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"]
    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

Nuovo dalla versione 5.0.

Non supportato dal cluster Elastic.

L'`$regexFind`operatore in Amazon DocumentDB viene utilizzato per eseguire la corrispondenza delle espressioni regolari su campi stringa all'interno dei documenti. Consente di cercare ed estrarre sottostringhe specifiche che corrispondono a un determinato modello di espressione regolare.

Parametri

- `input`: Il campo o l'espressione di stringa da cercare.
- `regex`: Il modello di espressione regolare da abbinare.
- `options`: (opzionale) Un oggetto che specifica parametri opzionali per l'espressione regolare, come la distinzione tra maiuscole e minuscole e la corrispondenza su più righe. Le opzioni supportate sono `i` (senza distinzione tra maiuscole e minuscole) e (multilinea). `m`

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$regexFind`operatore per cercare documenti in cui il `name` campo corrisponde a uno specifico modello di espressione regolare.

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.users.aggregate([
  {
    $project: {
```

```
      names: {
        $regexFind: { input: '$name', regex: 'j', options: 'i' }
      }
    }
  },
  { $match: {names: {$ne: null}}}
])
```

Questa query restituirà tutti i documenti in cui il `name` campo contiene la lettera «j» (senza distinzione tra maiuscole e minuscole).

Output

```
[
  { _id: 1, names: { match: 'J', idx: 0, captures: [] } }
]
```

Nota: se la tua query utilizza Amazon DocumentDB planner versione 1, devi usare un suggerimento per utilizzare un indice. Senza un suggerimento, la query può eseguire una scansione della raccolta. Per controllare la versione del planner in uso e saperne di più sull'uso dei suggerimenti, consulta. Pianificatore di query v2

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$regexFind` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

$ espressione regolare FindAll

Introdotto nella 5.0

L'`$regexFindAll`operatore in Amazon DocumentDB viene utilizzato per eseguire la corrispondenza delle espressioni regolari su campi stringa all'interno dei documenti. Consente di cercare ed estrarre sottostringhe specifiche che corrispondono a un determinato modello di espressione regolare, restituendo tutte le corrispondenze dell'espressione regolare.

Parametri

- `input`: Il campo o l'espressione di stringa da cercare.

- `regex`: Il modello di espressione regolare da abbinare.
- `options`: (opzionale) Un oggetto che specifica parametri opzionali per l'espressione regolare, come la distinzione tra maiuscole e minuscole e la corrispondenza su più righe. Le opzioni supportate sono `i` (senza distinzione tra maiuscole e minuscole) e (multilinea). `m`

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$regexFindAll`operatore per estrarre tutte le sequenze di lettere dal campo. `email`

Crea documenti di esempio

```
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" }

]);
```

Esempio di query

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      emailMatches: {
        $regexFindAll: { input: '$email', regex: '[a-z]+', options: 'i' }
      }
    }
  }
])
```

Output

```
[
  {
    _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: [] }
    ]
  }
]
```

Nota: se la tua query utilizza Amazon DocumentDB planner versione 1, devi usare un suggerimento per utilizzare un indice. Senza un suggerimento, la query può eseguire una scansione della raccolta. Per controllare la versione del planner in uso e saperne di più sull'uso dei suggerimenti, consulta. Pianificatore di query v2

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$regexFindAll` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

Node.js

Ecco un esempio di utilizzo dell'`$regexFind`operatore in un' Node.js applicazione:

```
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

Ecco un esempio di utilizzo dell'`$regexFind`operatore in un'applicazione 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([
```

```
    {
        "$project": {
            "name": 1,
            "emailMatches": {
                "$regexFindAll": {
                    "input": "$email",
                    "regex": "[a-z]+",
                    "options": "i"
                }
            }
        }
    }
]))

print(results)

client.close()
```

$RegexMatch

Nuovo dalla versione 5.0. Non supportato dal cluster Elastic.

L'`$regexMatch`operatore in Amazon DocumentDB viene utilizzato per eseguire la corrispondenza delle espressioni regolari sui campi stringa. Restituisce un valore booleano (`trueofalse`) che indica se la stringa di input corrisponde al modello specificato.

Parametri

- `input`: La stringa da testare rispetto all'espressione regolare.
- `regex`: Il modello di espressione regolare da abbinare.
- `options`: (Facoltativo) Contrassegni per modificare il comportamento delle espressioni regolari, ad esempio matching senza distinzione tra maiuscole e minuscole (`i`) o la corrispondenza multilinea (). `m`

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$regexMatch`operatore per verificare se i nomi iniziano con la lettera 'M'. L'operatore restituisce `true` or `false` per ogni documento.

Crea documenti di esempio

```
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" },

]);
```

Esempio di interrogazione

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      startsWithM: {
        $regexMatch: {
          input: "$name",
          regex: "^M",
          options: "i"
        }
      }
    }
  }
]);
```

Output

```
{ _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 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$regexMatch` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$ Sostituisci tutto

Introdotto nella 5.0

L'`$replaceAll`operatore in Amazon DocumentDB viene utilizzato per sostituire tutte le occorrenze di un modello di stringa specificato all'interno di un campo con una nuova stringa. Questo operatore può essere utile per attività come la normalizzazione dei dati, la pulizia del testo e la manipolazione delle stringhe.

Parametri

- `input`: il campo o l'espressione contenente la stringa da sostituire.
- `find`: Lo schema di stringhe da cercare e sostituire.
- `replacement`: La stringa con cui sostituire le occorrenze corrispondenti.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$replaceAll`operatore in una pipeline di aggregazione per sostituire tutte le occorrenze della stringa «Chocolatier» con «Chocolate Co.» nel campo «BrandName» di una collezione «products».

Crea documenti di esempio

```
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
  }
]);
```

Esempio di interrogazione

```
db.products.aggregate([
  {
    $addFields: {
      "brandName": {
        $replaceAll: {
          input: "$brandName",
          find: "Chocolatier",
          replacement: "Chocolate Co."
        }
      }
    }
  }
```

```
])
```

Output

```
[
  {
    _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
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$replaceAll` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$ Sostituisci uno

Introdotto nella 5.0

L'`$replaceOne`operatore in Amazon DocumentDB è un operatore di espressione di stringa utilizzato nelle pipeline di aggregazione per sostituire la prima occorrenza di una sottostringa specificata all'interno di una stringa con una stringa sostitutiva. Questo operatore fa distinzione tra maiuscole e minuscole e sostituisce solo la prima corrispondenza trovata.

Parametri

- `input`: La stringa (campo) su cui eseguire la ricerca.
- `find`: La stringa da cercare all'interno dell'input.
- `replacement`: La stringa che sostituisce la prima occorrenza della ricerca nell'input (campo).

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$replaceOne`operatore all'interno di una pipeline di aggregazione per sostituire le sottostringhe nei nomi dei prodotti.

Crea documenti di esempio

```
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 }
```

```
]);
```

Esempio di aggregazione

```
db.products.aggregate([
  {
    $addFields: {
      standardizedName: {
        $replaceOne: {
          input: "$name",
          find: "Pack",
          replacement: "Package"
        }
      }
    }
  }
]);
```

Output

L'output mostra che solo la prima occorrenza di «Pack» in ogni nome di prodotto è stata sostituita con «Package».

```
[
  {
    _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"
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$replaceOne` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$replaceRoot`operatore viene utilizzato per sostituire il documento radice con il documento incorporato specificato. Ciò è utile quando si desidera promuovere un documento annidato al livello superiore o ristrutturare l'output dei dati.

Parametri

- `newRoot`: il nuovo documento radice che sostituirà il documento radice esistente.

Esempio (MongoDB Shell)

Questo esempio mostra come estrarre le informazioni sull'indirizzo di spedizione dagli ordini dei clienti, utile per generare etichette di spedizione o elenchi di indirizzi.

Crea documenti di esempio

```
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 } ])
```

Esempio di interrogazione

```
db.orders.aggregate([
  {
    $replaceRoot: {
      newRoot: "$shippingAddress"
    }
  }
])
```

Output

```
{
    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'
  }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$replaceRoot` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$sostituisci con

Nuovo dalla versione 8.0

Non supportato dal cluster Elastic.

La fase di `$replaceWith` aggregazione in Amazon DocumentDB viene utilizzata per sostituire il documento di input con un nuovo documento. Tutti i campi esistenti nel documento di input, incluso il campo _id, vengono sostituiti dal nuovo documento. `$replaceWith`è comunemente usato per appiattire i documenti o promuovere un documento incorporato al livello superiore.

Parametri

- `<replacement>`(obbligatorio): Il nuovo documento che sostituirà il documento esistente.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$replaceWith`operatore per sostituire un documento esistente in una raccolta Amazon DocumentDB.

Crea documenti di esempio

```
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 ]
  }
]);
```

Esempio di interrogazione

```
db.restaurants.aggregate([
  { $replaceWith: {
      name: "$name",
      cuisine: "$cuisine",
      rating: { $avg: "$ratings" }
    }
  }
]);
```

Output

```
[
  {
    name: 'Biryani Adda',
    cuisine: 'Indian',
    rating: 3.4
  },
  {
    name: 'The Burger Spot',
    cuisine: 'American',
    rating: 2.7777777777777777
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$replaceWith` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$reverseArray`operatore in Amazon DocumentDB viene utilizzato per invertire gli elementi di un array nell'ordine specificato. Questo operatore è utile quando è necessario riordinare gli elementi di un array nella direzione inversa.

Parametri

- `expression`: L'espressione dell'array da invertire.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$reverseArray`operatore per invertire l'ordine degli elementi in un array.

Crea documenti di esempio

```
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]}
]);
```

Esempio di interrogazione

```
db.miles.aggregate([
  {
    $project: {
      _id: 1,
      member_since: 1,
      credit_card: 1,
      reversed_flight_miles: { $reverseArray: "$flight_miles" }
    }
  }
]);
```

Output

```
{ "_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 questo esempio, l'`$reverseArray`operatore viene utilizzato per invertire l'ordine dell'`flight_miles`array. Il `reversed_flight_miles` campo risultante nell'output mostra gli elementi dell'array nell'ordine inverso.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$reverseArray` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

Node.js

Ecco un esempio di utilizzo dell'`$reverseArray`operatore in un' Node.js applicazione:

```
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

Ecco un esempio di utilizzo dell'`$reverseArray`operatore in un'applicazione Python:

```
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()
```

$round

Nuovo dalla versione 8.0.1.

L'`$round`operatore in Amazon DocumentDB arrotonda un numero a una cifra decimale specificata. Quando un valore cade esattamente a metà strada tra i due valori più vicini in quel punto, viene `$round` arrotondato alla cifra pari più vicina. Ad esempio, 2,5 viene arrotondato a 2 e 3,5 viene arrotondato a 4.

Parametri

- `number`: Un'espressione che si risolve in un numero.
- `place`: facoltativo. Un'espressione intera compresa tra -20 e 100 che specifica il numero di cifre decimali a cui arrotondare. Un valore negativo viene arrotondato a sinistra del decimale. Il valore predefinito è 0.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$round`operatore per arrotondare i valori numerici a una cifra decimale.

Arrotondamento con un valore posizionale positivo

```
db.measurements.insertMany([
  {_id: 1, value: 3.456},
  {_id: 2, value: 7.891},
  {_id: 3, value: 12.345}
]);
```

Esempio di interrogazione

```
db.measurements.aggregate([
  { $project: { rounded: { $round: ["$value", 1] } } }
]);
```

Output

```
[
  {_id: 1, rounded: 3.5},
  {_id: 2, rounded: 7.9},
  {_id: 3, rounded: 12.3}
]
```

Arrotondare i valori a metà alla cifra pari più vicina

Quando un valore è esattamente a metà strada tra i due valori più vicini nel punto di destinazione, viene arrotondato alla cifra pari più vicina `$round` anziché arrotondare sempre per eccesso. L'esempio seguente arrotonda a metà i valori interi.

```
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] } } }
]);
```

Output

```
[
  {_id: 1, rounded: 10},
  {_id: 2, rounded: 12},
  {_id: 3, rounded: 12},
  {_id: 4, rounded: 14}
]
```

Poiché `10.5` è esattamente a metà strada tra `10` e`11`, viene arrotondato al valore pari più vicino. `10` Analogamente, `11.5` arrotonda `12.5` entrambi al valore `12` pari più vicino e `13.5` arrotonda a. `14`

Arrotondamento con un valore posizionale negativo

Quando `place` è negativo, viene `$round` arrotondato a sinistra del decimale. L'esempio seguente arrotonda i valori utilizzando la prima cifra a sinistra del decimale.

```
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] } } }
]);
```

Output

```
[
  {_id: 1, rounded: 20},
  {_id: 2, rounded: 30},
  {_id: 3, rounded: 30}
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$round` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

Nuovo dalla versione 4.0.

Non supportato dal cluster Elastic.

L'`$rtrim`operatore in Amazon DocumentDB viene utilizzato per rimuovere i caratteri finali da una stringa. Per impostazione predefinita, rimuove gli spazi bianchi finali, ma è anche possibile specificare un set di caratteri da rimuovere passando l'argomento chars.

Parametri

- `input`: La stringa di input da cui rimuovere i caratteri di spaziatura finali.
- `chars`: (opzionale) Per rimuovere caratteri specifici.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$rtrim`operatore per rimuovere i caratteri specificati finali (» *») da una stringa.

Crea documenti di esempio

```
db.collection.insert([
  { "name": "John Doe*  ", "age": 30 },
  { "name": "Jane Smith  ", "age": 25 },
  { "name": "Bob Johnson", "age": 35 }
]);
```

Esempio di interrogazione

```
db.collection.aggregate([
  {
    $project: {
      _id: 0,
      name: { $rtrim: { input: "$name", chars: " *" } },
      age: 1
    }
  }
]);
```

Output

```
[
  { age: 30, name: 'John Doe' },
  { age: 25, name: 'Jane Smith' },
  { age: 35, name: 'Bob Johnson' }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$rtrim` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

La fase di `$sample` aggregazione in Amazon DocumentDB viene utilizzata per selezionare in modo casuale un numero specifico di documenti da una raccolta. Ciò è utile per attività come l'analisi dei dati, il test e la generazione di campioni per un'ulteriore elaborazione.

Parametri

- `size`: il numero di documenti da selezionare casualmente.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare lo `$sample` stage per selezionare casualmente due documenti dalla raccolta. `temp`

Create documenti di esempio

```
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") }
]);
```

Esempio di query

```
db.temp.aggregate([
   { $sample: { size: 2 } }
])
```

Output

```
{ "_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") }
```

Come mostrano i risultati, 2 dei 10 documenti sono stati campionati casualmente. È ora possibile utilizzare questi documenti per determinare una media o eseguire calcoli. min/max

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$sample` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$ricerca

Nuovo dalla versione 5.0.

L'`$search`operatore in Amazon DocumentDB viene utilizzato per fornire funzionalità di ricerca di testo.

Parametri

Nessuno

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$search`operatore per eseguire una query di ricerca testuale.

Crea documenti di esempio

```
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." }
]);
```

Esempio di interrogazione

```
db.textcollection.find(
  { $text: { $search: "John" } }
);
```

Output

```
[
  {
    _id: 1,
```

```
    name: 'John Doe',
    description: 'This is a sample document about John Doe.'
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$search` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$second`operatore in Amazon DocumentDB estrae il componente dei secondi da una data o da un timestamp. Viene utilizzato per recuperare il valore dei secondi da un campo di data o timestamp.

Parametri

- `expression`: Il campo data o timestamp da cui estrarre il valore dei secondi. Questa espressione può essere un percorso di campo o qualsiasi espressione valida che si risolve in una data o un timestamp.

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare l'`$second`operatore per estrarre il componente dei secondi da un campo data.

Crea documenti di esempio

```
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") }
]);
```

Esempio di interrogazione

```
db.users.aggregate([{ $project: { name: 1, dobSeconds: { $second: "$dob" } } }])
```

Output

```
[
  { "_id" : ObjectId("6089a9c306a829d1f8b456a1"), "name" : "John", "dobSeconds" : 45 },
```

```
  { "_id" : ObjectId("6089a9c306a829d1f8b456a2"), "name" : "Jane", "dobSeconds" : 59 },
  { "_id" : ObjectId("6089a9c306a829d1f8b456a3"), "name" : "Bob", "dobSeconds" : 0 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$second` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

Nuovo dalla versione 8.0

Non supportato da Elastic Cluster.

La fase di `$set` aggregazione in Amazon DocumentDB consente di aggiungere nuovi campi o aggiornare i valori dei campi esistenti nella documentazione durante una pipeline di aggregazione.

Parametri

- `expression`: L'espressione da valutare. Può essere qualsiasi espressione di aggregazione valida, inclusi riferimenti di campo e operazioni aritmetiche.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'uso della fase di `$set` aggregazione per calcolare i totali moltiplicando il campo per il campo. `quantity` `price`

Crea documenti di esempio

```
db.inventory.insertMany([
  { item: "pencil", quantity: 100, price: 0.24},
  { item: "pen", quantity: 204, price: 1.78 }
]);
```

Esempio di aggregazione

```
db.inventory.aggregate([
  {
    $set: {
      total: { $multiply: ["$quantity", "$price"] }
    }
  }
])
```

Output

```
[
  {
    _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
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$set` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

Novità dalla versione 4.0.

L'`$setDifference`operatore in Amazon DocumentDB viene utilizzato per confrontare due set e restituire gli elementi che si trovano nel primo set ma non nel secondo set. Questo operatore è utile per trovare gli elementi unici tra due set.

Parametri

- `firstSet`: Il primo set da confrontare.
- `secondSet`: Il secondo set da confrontare.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$setDifference`operatore per trovare gli elementi unici tra due set.

Crea documenti di esempio

```
db.collection.insertMany([
  { _id: 1, fruits: ["apple", "banana", "cherry", "date"] },
  { _id: 2, fruits: ["banana", "cherry", "date", "elderberry"] }
]);
```

Esempio di interrogazione

```
db.collection.aggregate([
  {
    $project: {
      uniqueFruits: { $setDifference: ["$fruits", ["banana", "cherry", "date"]] }
    }
  }
]);
```

Output

```
[
  { "_id": 1, "uniqueFruits": ["apple"] },
  { "_id": 2, "uniqueFruits": ["elderberry"] }
]
```

L'interrogazione esegue le seguenti operazioni:

1. Utilizza lo `$project` stage per creare un nuovo campo `uniqueFruits` per ogni documento.

2. L'`$setDifference`operatore confronta l'`fruits`array con l'array `["banana", "cherry", "date"]` e restituisce gli elementi unici dell'`fruits`array.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$setDifference` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

Node.js

Ecco un esempio di come utilizzare l'`$setDifference`operatore in un' Node.js applicazione:

```
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

Ecco un esempio di come utilizzare l'`$setDifference`operatore in un'applicazione 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['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

L'`$setEquals`operatore in Amazon DocumentDB viene utilizzato per determinare se due set sono uguali. Confronta due array e restituisce `true` se contengono gli stessi elementi distinti, indipendentemente dal loro ordine.

Parametri

- `expression1`: La prima matrice da confrontare.
- `expression2`: Il secondo array da confrontare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$setEquals`operatore per confrontare due set di valori.

Crea documenti di esempio

```
db.collection.insertMany([
  { _id: 1, fruits: ["apple", "banana", "cherry"] },
  { _id: 2, fruits: ["banana", "apple", "cherry"] },
  { _id: 3, fruits: ["apple", "banana", "orange"] }
])
```

Esempio di query

```
db.collection.find({
  $expr: {
    $setEquals: ["$fruits", ["apple", "banana", "cherry"]]
  }
})
```

Output

```
{ "_id" : 1, "fruits" : [ "apple", "banana", "cherry" ] }
{ "_id" : 2, "fruits" : [ "banana", "apple", "cherry" ] }
```

La query utilizza l'`$setEquals`operatore per confrontare il `fruits` campo di ogni documento con l'array`["apple", "banana", "cherry"]`. Vengono restituiti i documenti in cui il `fruits` campo è uguale all'array di confronto.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$setEquals` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$setIntersection`operatore in Amazon DocumentDB viene utilizzato per restituire gli elementi comuni tra due o più array. Questo operatore è particolarmente utile quando si lavora con set di dati e consente di trovare l'intersezione di più set.

Parametri

- `array1`: La prima matrice da intersecare.
- `array2`: Il secondo array da intersecare.
- `arrayN`: (opzionale) Matrici aggiuntive da intersecare.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$setIntersection`operatore per trovare gli elementi comuni tra due array.

Crea documenti di esempio

```
db.collection.insertMany([
  { _id: 1, colors: ["red", "blue", "green"] },
  { _id: 2, colors: ["blue", "yellow", "orange"] },
  { _id: 3, colors: ["red", "green", "purple"] }
])
```

Esempio di query

```
db.collection.aggregate([
  { $project: {
      _id: 1,
      commonColors: { $setIntersection: ["$colors", ["red", "blue", "green"]] }
    }
  }
])
```

Output

```
[
  { "_id": 1, "commonColors": ["red", "blue", "green"] },
  { "_id": 2, "commonColors": ["blue"] },
  { "_id": 3, "commonColors": ["red", "green"] }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$setIntersection` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$set IsSubset

L'`$setIsSubset`operatore in Amazon DocumentDB viene utilizzato per determinare se un set di valori è un sottoinsieme di un altro set. È utile per eseguire confronti e operazioni basati su set su campi di matrice.

Parametri

- `field`: Il campo a cui applicare l'`$setIsSubset`operatore.
- `set`: Il set con cui confrontare il campo.

Esempio (MongoDB Shell)

L'esempio seguente mostra l'utilizzo dell'`$setIsSubset`operatore per verificare se il `tags` campo è un sottoinsieme del set specificato.

Crea documenti di esempio

```
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"] }
]);
```

Esempio di interrogazione

```
db.products.find({
  $expr: { $setIsSubset: [["tag1", "tag2"], "$tags"] }
})
```

*Nota: * `$setIsSubset` è un operatore di aggregazione e non può essere utilizzato direttamente nelle query find (). In questo esempio, `$expr` viene utilizzato con `find()` per colmare il divario tra gli operatori di query e le espressioni di aggregazione.

Output

```
[
  { "_id" : 1, "name" : "Product A", "tags" : [ "tag1", "tag2", "tag3" ] },
  { "_id" : 2, "name" : "Product B", "tags" : [ "tag1", "tag2" ] }
]
```

La query restituisce i documenti in cui il `tags` campo è un sottoinsieme dell'insieme. `["tag1", "tag2"]`

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$setIsSubset` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'operatore di `$setUnion` aggregazione in Amazon DocumentDB viene utilizzato per combinare due o più set di valori e restituire un set che contiene tutti gli elementi unici dei set di input. Questo operatore è utile quando è necessario eseguire operazioni basate su set su campi array nei documenti.

Parametri

- `expression1`: espressione che si risolve in un array.
- `expression2`: espressione che si risolve in una matrice.
- `expressionN`: espressioni aggiuntive che si risolvono in matrici (opzionale).

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$setUnion`operatore per combinare gli elementi unici di due campi di matrice in una raccolta.

Crea documenti di esempio

```
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"] }
]);
```

Esempio di interrogazione

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      allInterests: { $setUnion: ["$hobbies", "$skills"] }
    }
  }
]);
```

Output

```
[
  { "_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 questo esempio, l'`$setUnion`operatore viene utilizzato per combinare gli elementi univoci dei campi `hobbies` e `skills` array per ogni documento utente. Il `allInterests` campo risultante contiene l'unione di tutti gli hobby e le abilità unici di ogni utente.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$setUnion` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

$set WindowFields

Nuovo dalla versione 8.0.2.

La `$setWindowFields` fase di Amazon DocumentDB esegue operazioni su un determinato intervallo di documenti (una finestra) in una partizione e restituisce i risultati in base all'operatore di finestra scelto. La fase raggruppa i documenti di input in partizioni, facoltativamente li ordina e per ogni documento calcola uno o più campi di output su una finestra di documenti circostanti.

Sintassi

```
{
  $setWindowFields: {
    partitionBy: <expression>,
    sortBy: { <sortField1>: <sortOrder>, ... },
    output: {
      <outputField1>: {
        <windowOperator>: <expression>,
        window: {
          documents: [ <lowerBound>, <upperBound> ],
          range: [ <lowerBound>, <upperBound> ],
```

```
          unit: <time unit>
        }
      },
      ...
    }
  }
}
```

Parametri

- `partitionBy`: facoltativo. Una singola espressione utilizzata per raggruppare i documenti in partizioni. Può essere un percorso di campo (ad esempio`"$state"`) o un'altra espressione. Se omesso, tutti i documenti di input appartengono a una singola partizione.
- `sortBy`: facoltativo. Un documento che specifica il campo o i campi in base ai quali ordinare i documenti di ciascuna partizione, in ordine crescente o decrescente. `{ <field>: 1 }` `{ <field>: -1 }` A `sortBy` è obbligatorio quando si utilizzano limiti di finestra basati su documenti o intervalli e quando si utilizza uno degli operatori di classificazione.
- `output`: richiesto. Un documento che specifica i campi da aggiungere a ciascun documento. È possibile specificare più campi di output, ma ogni campo di output richiede esattamente un operatore di finestra, il relativo argomento e un `window` documento opzionale che definisce i limiti della finestra:. `{ <field>: { <windowOperator>: <expression>, window: { ... } } }`

Limiti delle finestre

All'interno di ogni `output` campo è possibile definire facoltativamente i confini della finestra con un `window` documento. Una finestra può essere definita in base alla posizione del documento o a un intervallo di valori.

- `documents`: specifica la finestra come numero di documenti prima e dopo il documento corrente, nel modulo`[ <lowerBound>, <upperBound> ]`. Ogni limite è un offset intero (ad esempio, `-1` o2) o una delle parole chiave and. `"unbounded"` `"current"` `"unbounded"`estende la finestra all'inizio della partizione se utilizzata come limite inferiore o alla fine della partizione se utilizzata come limite superiore. `"current"`indica il documento corrente. Una `documents` finestra richiede un `sortBy` a meno che non lo siano entrambi i `"unbounded"` limiti.
- `range`: specifica la finestra come un intervallo di valori di `sortBy` campo relativi al documento corrente, nel modulo. `[ <lowerBound>, <upperBound> ]` Ogni limite è un numero che viene aggiunto al `sortBy` valore del documento corrente o alla `"unbounded"` parola chiave. Una `range`

finestra richiede esattamente un `sortBy` campo crescente e non accetta la `"current"` parola chiave.

- `unit`: facoltativo. Utilizzato con `range` per interpretare i limiti numerici della finestra come un intervallo di tempo in cui il `sortBy` campo contiene valori di data. Amazon DocumentDB supporta le seguenti unità:`"year"`,,`"quarter"`,,`"month"`,`"week"`, `"day"``"hour"`, `"minute"` e. `"second"` `"millisecond"` Quando `unit` è specificato, i `range` limiti devono essere numeri interi.

Quando il `window` documento viene omesso, la finestra predefinita è`documents: ["unbounded", "unbounded"]`, che copre l'intera partizione.

> **Note**
>
> Quando si utilizza una `range` finestra sui valori di data, si consiglia una durata fissa `unit` come `"day"` (o inferiore) anziché`"month"`, o `"quarter"``"year"`, le cui lunghezze dipendenti dal calendario possono generare limiti ambigui della finestra.

Operatori di finestra supportati

Amazon DocumentDB supporta i seguenti 26 operatori di finestre nel `output` campo dello `$setWindowFields` stage:

- `$sum`
- `$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**
>
> L'`$rand`operatore non è supportato all'interno di un operatore di finestra nello `$setWindowFields` stage.

Finestre vuote

Quando una finestra non contiene documenti (ad esempio, una `range` finestra senza valori nell'intervallo), il risultato dipende dall'operatore: `$count` and `$sum` return`0`;`$push`,,,`$addToSet`,`$firstN`, `$lastN` `$minN` `$maxN``$topN`, e `$bottomN` restituisce un array vuoto; `$percentile` restituisce un array di `null` valori (uno per percentile richiesto); tutti gli altri operatori restituiscononu`ll`.

Esempio (MongoDB Shell)

L'esempio seguente partiziona i documenti in base`region`, ordina ogni partizione in base `day` e calcola un totale progressivo di `revenue` dall'inizio della partizione fino al documento corrente.

Create documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.sales.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$region",
      sortBy: { day: 1 },
      output: {
        runningRevenue: {
          $sum: "$revenue",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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 }
]
```

Ogni documento è `runningRevenue` arricchito dalla somma cumulativa della `revenue` sua partizione fino al documento corrente incluso.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dello `$setWindowFields` stage, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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()
```

$sigmoide

Nuovo dalla versione 8.0.1.

L'`$sigmoid`operatore in Amazon DocumentDB applica la funzione sigmoide logistica a un input numerico`1 / (1 + e^(-x))`. La funzione mappa qualsiasi numero reale su un valore compreso tra 0 e 1, producendo una curva. S-shaped Ciò è utile quando è necessario normalizzare i valori nell'intervallo 0-1, ad esempio convertire i punteggi grezzi in punteggi di probabilità o di pertinenza.

Parametri

- `expression`: Un'espressione che si risolve in un valore numerico.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$sigmoid`operatore per convertire i valori logit in probabilità.

Crea documenti di esempio

```
db.predictions.insertMany([
  {_id: 1, logit: -2},
  {_id: 2, logit: 0},
  {_id: 3, logit: 2}
]);
```

Esempio di interrogazione

```
db.predictions.aggregate([
  { $project: { probability: { $sigmoid: "$logit" } } }
]);
```

Output

```
[
  {_id: 1, probability: 0.11920292202211755},
  {_id: 2, probability: 0.5},
  {_id: 3, probability: 0.8807970779778823}
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$sigmoid` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

Nuovo dalla versione 8.0.1.

L'`$sin`operatore in Amazon DocumentDB restituisce il seno di un valore misurato in radianti. Utilizzalo nella pipeline di aggregazione per eseguire calcoli trigonometrici su campi numerici.

Parametri

- `expression`: Un'espressione che si risolve in un numero in radianti.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$sin`operatore per calcolare il seno dei valori angolari in radianti.

Crea documenti di esempio

```
db.angles.insertMany([
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.5235987755982988 },
  { "_id": 3, "angle": 1.5707963267948966 }
]);
```

Esempio di interrogazione

```
db.angles.aggregate([
  { $project: {
    "sine": { $sin: "$angle" }
  }}
]);
```

Output

```
[
  { "_id": 1, "sine": 0 },
```

```
  { "_id": 2, "sine": 0.49999999999999994 },
  { "_id": 3, "sine": 1 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$sin` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

Nuovo dalla versione 8.0.1.

L'`$sinh`operatore in Amazon DocumentDB restituisce il seno iperbolico di un valore. Utilizzalo nella pipeline di aggregazione per eseguire calcoli trigonometrici iperbolici su campi numerici.

Parametri

- `expression`: Un'espressione che si risolve in un numero.

Il tipo restituito è predefinito`double`. Se l'input è un decimale a 128 bit, anche l'output è un decimale a 128 bit.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$sinh`operatore per calcolare il seno iperbolico dei valori numerici.

Crea documenti di esempio

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Esempio di interrogazione

```
db.values.aggregate([
  { $project: {
    "result": { $sinh: "$value" }
  }}
]);
```

Output

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 1.1752011936438014 },
  { "_id": 3, "result": -1.1752011936438014 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$sinh` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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, l'`$skip`operatore viene utilizzato per compensare il punto iniziale dei risultati della query, permettendo di recuperare un sottoinsieme specifico dei documenti corrispondenti. Ciò è particolarmente utile negli scenari di impaginazione, in cui si desidera recuperare pagine successive di risultati.

Parametri

- `skip`: Il numero di documenti da saltare prima di restituire i documenti rimanenti.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$skip`operatore per recuperare la seconda pagina di risultati (documenti 11-20) da una raccolta.

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.collection.find({}, { "name": 1 })
             .skip(10)
             .limit(10);
```

Output

```
[
  { "_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" }
```

```
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$skip` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'operatore di `$slice` aggregazione consente di restituire un sottoinsieme di un array attraversando l'array dall'inizio o dalla fine dell'array. Viene utilizzato per visualizzare un numero limitato di elementi da un campo di matrice, ad esempio gli N elementi superiori o inferiori.

Parametri

- `array`: Il campo dell'array da suddividere.
- n: un numero intero che specifica il numero di elementi da restituire. Un valore positivo inizia dall'inizio dell'array, mentre un valore negativo inizia dalla fine dell'array.

Esempio (MongoDB Shell)

L'esempio seguente mostra come `$slice` restituire i primi due dolci preferiti per ogni chef.

Crea documenti di esempio

```
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" ] }
]);
```

Esempio di interrogazione

```
db.sweets.aggregate([
```

```
  { $project: { _id: 0, name: 1, topTwoFavorites: { $slice: [ "$favorites", 2 ] } } }
]);
```

Output

```
[
  { name: 'Alvin', topTwoFavorites: [ 'chocolate', 'cake' ] },
  { name: 'Tom', topTwoFavorites: [ 'donuts', 'pudding' ] },
  { name: 'Jessica', topTwoFavorites: [ 'fudge', 'smores' ] },
  { name: 'Rachel', topTwoFavorites: [ 'ice cream' ] }
]
```

In questo esempio, l'`$slice`operatore viene utilizzato per estrarre i primi due elementi dall'`favorites`array per ogni documento.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$slice` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$size`operatore viene utilizzato per restituire il conteggio degli elementi all'interno di un campo di matrice. Questo può essere usato per determinare il numero di elementi in un array archiviato in un documento.

Parametri

- `field`: il percorso del campo di cui si desidera restituire la dimensione dell'array.

Esempio (MongoDB Shell)

Questo esempio mostra come utilizzare l'`$size`operatore per restituire il numero di team seguiti da ciascun utente.

Crea documenti di esempio

```
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"] }
]);
```

Esempio di interrogazione

```
db.profiles.aggregate([
  {
    $project: {
      _id: 0,
      name: 1,
      "numberOfTeams": { $size: "$teams" }
    }
  }
])
```

Output

```
[
  { name: 'John Doe', numberOfTeams: 3 },
  { name: 'Jane Smith', numberOfTeams: 2 },
  { name: 'Bob Johnson', numberOfTeams: 3 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$size` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

La fase di `$sort` aggregazione ordina i documenti nella pipeline in base a valori di campo specificati. I documenti sono disposti in ordine crescente o decrescente in base ai criteri di ordinamento forniti.

Parametri

- `field`: il nome del campo in base al quale ordinare.
- `order`: utilizzare 1 per l'ordine crescente o `-1` per l'ordine decrescente.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo della `$sort` fase per ordinare i prodotti in base al prezzo in ordine decrescente.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.products.aggregate([
  { $sort: { price: -1 } }
]);
```

Output

```
[
  { _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 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo della fase di `$sort` aggregazione, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

Ordinamento incrementale

L'ordinamento incrementale è un'ottimizzazione in cui Amazon DocumentDB utilizza un indice che ordina già i campi principali di un ordinamento multicampo. Amazon DocumentDB ordina quindi solo i campi rimanenti all'interno di ciascun gruppo di documenti che condividono gli stessi valori iniziali. Invece di ordinare l'intero set di risultati in memoria come un blocco (ordinamento completo), Amazon DocumentDB legge i documenti in ordine di indice. Quindi esegue molti piccoli ordinamenti indipendenti, uno per gruppo. I risultati sono identici a quelli di un ordinamento completo. Cambiano solo il piano di interrogazione, la quantità di memoria e il lavoro utilizzati.

Vantaggi

L'ordinamento incrementale offre i seguenti vantaggi:

- Minore utilizzo della memoria e minori perdite di ordinamento basate su disco. Amazon DocumentDB ordina un piccolo gruppo alla volta anziché l'intero set di risultati contemporaneamente.
- Risultati più rapidi con**`limit`**. Amazon DocumentDB può emettere documenti ordinati prima di leggere l'intero input. Un ordinamento seguito da a `limit` (ad esempio, una query Top-N o impaginata) può essere interrotto non appena viene prodotto un numero sufficiente di documenti.

Quando Amazon DocumentDB sceglie un ordinamento incrementale

Ad esempio, se l'ordinamento è attivo `{ customerId: 1, orderDate: -1 }` e l'indice è attivo`{ customerId: 1 }`, l'indice restituisce già i documenti ordinati per. `customerId` Amazon DocumentDB deve solo ordinare ogni gruppo di documenti che condividono lo stesso criterio`customerId`. `orderDate` Il campo principale (`customerId`) già ordinato è la chiave preordinata. A seconda del tipo di indice già fornito, Amazon DocumentDB sceglie uno dei tre comportamenti:

- L'indice ordina ogni chiave di ordinamento. Non è necessario alcun ordinamento. Amazon DocumentDB legge i documenti direttamente nell'ordine richiesto.
- L'indice ordina un prefisso iniziale delle chiavi di ordinamento. Amazon DocumentDB utilizza un ordinamento incrementale, ordinando solo le chiavi finali all'interno di ciascun gruppo preordinato.
- Nessun indice ordina alcuna chiave di ordinamento iniziale. Amazon DocumentDB esegue un ordinamento completo sull'intero input.

Requisiti e limitazioni

L'ordinamento incrementale richiede Amazon DocumentDB 8.0.2 o versioni successive, con Planner versione 3.0 (il pianificatore di query predefinito di Amazon DocumentDB 8.0.2). Per ulteriori informazioni sulle versioni del planner, consulta. Pianificatore di query v3

Amazon DocumentDB non può utilizzare un indice parziale o un indice a più chiavi (array) per fornire il prefisso preordinato per un ordinamento incrementale. In questi casi, Amazon DocumentDB ricorre a un ordinamento completo. Ciò non influisce mai sulla correttezza dei risultati.

Verifica di un ordinamento incrementale con **`explain()`**

`explain()`Da utilizzare per confermare che una query utilizza un ordinamento incrementale. Nel piano vincente, un ordinamento incrementale appare come una `SORT` fase che include una `presortedKeys` matrice che elenca i campi di ordinamento principali già ordinati nell'indice. Un ordinamento completo appare come una `SORT` fase senza `presortedKeys` matrice. Il `presortedKeys` campo è un' DocumentDB-specific aggiunta di Amazon all'`explain()`output e non viene visualizzato nei piani di spiegazione di 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

Nuovo dalla versione 8.0

Non supportato dal cluster Elastic.

L'`$sortArray`operatore in Amazon DocumentDB ordina gli elementi di un array. Per gli array di valori scalari, l'array viene ordinato per valore dell'elemento. Per gli array di documenti, l'array viene ordinato in base al campo o ai campi specificati. L'operatore restituisce l'array ordinato in ordine crescente (1) o decrescente (). `-1`

> **Note**
>
> `$sortArray`è supportato nella `$project` fase della pipeline di aggregazione con la versione 3 di Query Planner.

Sintassi

```
{ $sortArray: { input: <array>, sortBy: <sort specification> } }
```

Parametri

- `input`: L'array da ordinare.
- `sortBy`: Il criterio di ordinamento. Per una matrice di valori scalari, specificare 1 se è crescente o `-1` decrescente. Per una serie di documenti, specificate un documento del modulo `{ field: 1 }` o `{ field: -1 }` per ogni campo in base al quale ordinare.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$sortArray`operatore per ordinare gli elementi di un array in ordine crescente.

Crea documenti di esempio

```
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]}
]);
```

Esempio di interrogazione

```
db.miles.aggregate([
  {
    $project: {
      _id: 1,
      member_since: 1,
      credit_card: 1,
      sorted_flight_miles: { $sortArray: { input: "$flight_miles", sortBy: 1 } }
    }
  }
]);
```

Output

```
{ "_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 questo esempio, l'`$sortArray`operatore ordina l'`flight_miles`array in ordine crescente. Il `sorted_flight_miles` campo risultante nell'output mostra gli elementi dell'array in ordine ordinato.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$sortArray` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

Node.js

Ecco un esempio di utilizzo dell'`$sortArray`operatore in un' Node.js applicazione:

```
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

Ecco un esempio di utilizzo dell'`$sortArray`operatore in un'applicazione Python:

```
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()
```

$ordina ByCount

Nuovo dalla versione 8.0.1.

La fase di `$sortByCount` aggregazione in Amazon DocumentDB raggruppa i documenti in arrivo in base al valore di un'espressione specificata, quindi calcola il numero di documenti in ogni gruppo distinto e ordina i risultati in base al numero in ordine decrescente. È equivalente a una `$group` fase con un `$sum` accumulatore seguita da una fase nel campo del conteggio. `$sort`

Parametri

- `expression`: L'espressione per cui raggruppare. Può essere un percorso di campo (preceduto da$) o qualsiasi espressione di aggregazione valida.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare lo `$sortByCount` stage per contare e ordinare i documenti in base a un campo di categoria.

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.products.aggregate([
  { $sortByCount: "$category" }
```

```
]);
```

Output

```
[
  { "_id": "electronics", "count": 3 },
  { "_id": "food", "count": 2 },
  { "_id": "tools", "count": 1 }
]
```

I risultati vengono ordinati automaticamente `count` in ordine decrescente.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$sortByCount` dello stage, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'operatore di `$split` aggregazione in Amazon DocumentDB viene utilizzato per suddividere una stringa in un array di sottostringhe, in base a un delimitatore specificato. Questo può essere utile per analizzare campi di stringhe complessi ed estrarre singoli componenti per un'ulteriore elaborazione.

Parametri

- `string`: La stringa da dividere.
- `delimiter`: Il carattere o la stringa usata per dividere la stringa di input.

Esempio (MongoDB Shell)

In questo esempio, separiamo `$split` i componenti di un campo «Desk» in un array, semplificando l'elaborazione dei dati.

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.people.aggregate([
  { $project: { parts: { $split: ["$Desk", "-"] } } }
]);
```

Output

```
{ "_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" ] }
```

L'output di `$split` crea un array che può essere utilizzato nell'applicazione per visualizzare le informazioni per i dipendenti.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$split` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

Novità dalla versione 4.0.

L'`$sqrt`operatore in Amazon DocumentDB viene utilizzato per calcolare la radice quadrata di un numero.

Parametri

- `expression`: L'argomento può essere qualsiasi espressione valida purché si risolva in un numero non negativo.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$sqrt`operatore per calcolare la radice quadrata di un numero.

Crea documenti di esempio

```
db.numbers.insertMany([
  { "_id": 1, "number": 16 },
  { "_id": 2, "number": 36 },
  { "_id": 3, "number": 64 }
]);
```

Esempio di interrogazione

```
db.numbers.aggregate([
  { $project: {
    "_id": 1,
    "square_root": { $sqrt: "$number" }
  }}
]);
```

Output

```
[
  { _id: 1, square_root: 4 },
  { _id: 2, square_root: 6 },
  { _id: 3, square_root: 8 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$sqrt` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$standard DevPop

Nuovo dalla versione 8.0.1.

L'`$stdDevPop`operatore in Amazon DocumentDB calcola la deviazione standard dei valori numerici della popolazione. Come accumulatore, calcola la deviazione standard della popolazione tra i documenti all'interno di un gruppo nella fase di una pipeline di aggregazione. `$group` Come espressione, calcola la deviazione standard della popolazione di una serie di numeri. La deviazione standard della popolazione utilizza N come divisore (non). N-1 Non-numeric i valori vengono ignorati. Se non sono presenti valori numerici, viene restituito. `null` Se è presente un solo valore numerico, viene restituito. `0`

Parametri

- `expression`: un'espressione che si risolve in un valore numerico o in una matrice di valori numerici.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$stdDevPop`operatore per calcolare la deviazione standard della popolazione dei punteggi per soggetto.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.scores.aggregate([
  { $group: {
      _id: "$subject",
      stdDev: { $stdDevPop: "$score" }
    }}
]);
```

Output

```
[
  { "_id": "math", "stdDev": 10.75174404457249 },
  { "_id": "science", "stdDev": 12.441864811996632 }
]
```

Esempio di utilizzo delle espressioni (MongoDB Shell)

L'`$stdDevPop`operatore può essere utilizzato anche come espressione all'interno di uno `$project` stadio per calcolare la deviazione standard della popolazione di un campo di matrice.

Crea documenti di esempio

```
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] }
]);
```

Esempio di interrogazione

```
db.experiments.aggregate([
  { $project: {
      stdDev: { $stdDevPop: "$measurements" }
    }}
]);
```

Output

```
[
  { "_id": 1, "stdDev": 2.8284271247461903 },
  { "_id": 2, "stdDev": 0 },
  { "_id": 3, "stdDev": 2.8284271247461903 }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$stdDevPop`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce la deviazione standard della popolazione dell'espressione specificata per i documenti in ciascuna finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

Create documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente partiziona i documenti in base`sensor`, ordina ogni partizione in base `time` e restituisce la deviazione standard della popolazione corrente `reading` compresa tra l'inizio della partizione e il documento corrente.

```
db.sensorData.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$sensor",
      sortBy: { time: 1 },
      output: {
        runningStdDev: {
          $stdDevPop: "$reading",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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 }
]
```

Ogni documento è arricchito con `runningStdDev` la deviazione standard della popolazione `reading` all'interno della sua partizione fino al documento corrente incluso. Quando la finestra contiene un singolo valore, restituisce. `$stdDevPop` `0`

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$stdDevPop` dell'operatore, scegliete la scheda corrispondente alla lingua che desiderate utilizzare. Gli esempi seguenti mostrano l'uso dell'accumulatore (in`$group`), l'uso delle espressioni (in`$project`) e l'uso dell'operatore di finestra (in`$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()
```

$standard DevSamp

Nuovo dalla versione 8.0.1.

L'`$stdDevSamp`operatore in Amazon DocumentDB calcola la deviazione standard del campione di valori numerici. Come accumulatore, calcola la deviazione standard del campione tra i documenti all'interno di un gruppo nella fase di una pipeline di aggregazione. `$group` Come espressione, calcola la deviazione standard del campione di una serie di numeri. La deviazione standard del campione viene utilizzata N-1 come divisore (correzione di Bessel). Non-numeric i valori vengono ignorati. Se sono presenti meno di due valori numerici, viene restituito. `null`

Parametri

- `expression`: un'espressione che si risolve in un valore numerico o in una matrice di valori numerici.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$stdDevSamp`operatore per calcolare la deviazione standard del campione dei punteggi per soggetto.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.scores.aggregate([
  { $group: {
      _id: "$subject",
      stdDev: { $stdDevSamp: "$score" }
    }}
]);
```

Output

```
[
  { "_id": "math", "stdDev": 6.454972243679028 },
  { "_id": "science", "stdDev": 6.454972243679028 }
]
```

Esempio di utilizzo delle espressioni (MongoDB Shell)

L'`$stdDevSamp`operatore può essere utilizzato anche come espressione all'interno di uno `$project` stadio per calcolare la deviazione standard del campione di un campo di matrice.

Crea documenti di esempio

```
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] }
]);
```

Esempio di interrogazione

```
db.experiments.aggregate([
  { $project: {
      stdDev: { $stdDevSamp: "$measurements" }
    }}
]);
```

Output

```
[
  { "_id": 1, "stdDev": 3.1622776601683795 },
  { "_id": 2, "stdDev": 0 },
  { "_id": 3, "stdDev": 3.1622776601683795 }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$stdDevSamp`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce la deviazione standard di esempio dell'espressione specificata per i documenti in ciascuna finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

Create documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente partiziona i documenti in basesensor, ordina ogni partizione in base time e restituisce la deviazione standard del campione corrente value dall'inizio della partizione al documento corrente.

```
db.sensorReadings.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$sensor",
      sortBy: { time: 1 },
      output: {
        runningStdDev: {
          $stdDevSamp: "$value",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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 }
]
```

Ogni documento è integrato con runningStdDev la deviazione standard di esempio all'valueinterno della sua partizione fino al documento corrente incluso. Quando la finestra contiene meno di due valori (il primo documento in ogni partizione), ritorna. $stdDevSamp null

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo $stdDevSamp dell'operatore, scegliete la scheda corrispondente alla lingua che desiderate utilizzare. Gli esempi seguenti mostrano l'uso dell'accumulatore (in$group), l'uso delle espressioni (in$project) e l'uso dell'operatore di finestra (in$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()
```

$star LenBytes

L'`$strLenBytes`operatore in Amazon DocumentDB viene utilizzato per determinare la lunghezza di una stringa in byte. Ciò è utile quando è necessario comprendere la dimensione di archiviazione di un campo di stringa, specialmente quando si tratta di caratteri Unicode che possono utilizzare più di un byte per carattere.

Parametri

- `expression`: L'espressione stringa di cui calcolare la lunghezza.

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'`$strLenBytes`operatore per calcolare la lunghezza dei campi di stringa in byte.

Crea documenti di esempio

```
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" }
]);
```

Esempio di query

```
db.people.aggregate([
  {
    $project: {
      "Desk": 1,
      "length": { $strLenBytes: "$Desk" }
    }
  }
])
```

Output

```
{ "_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 }
```

Si noti che la lunghezza della stringa «Düsseldorf-BVV-021» è di 19 byte, che è diversa dal numero di punti di codice (18) a causa del carattere Unicode «Ü» che occupa 2 byte.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$strLenBytes` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$strLenCP`operatore in Amazon DocumentDB viene utilizzato per determinare la lunghezza di un'espressione stringa in punti di codice (caratteri Unicode). Ciò è utile quando è necessario conoscere il numero di caratteri in una stringa, anziché il numero di byte.

Parametri

- `expression`: L'espressione stringa per la quale restituire la lunghezza in punti di codice.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$strLenCP`operatore per determinare la lunghezza delle stringhe con caratteri Unicode.

Crea documenti di esempio

```
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" }
])
```

Esempio di interrogazione

```
db.people.aggregate([
  {
    $project: {
      "Desk": 1,
      "length": { $strLenCP: "$Desk" }
    }
  }
```

```
])
```

Output

```
{ "_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 }
```

Notate la differenza nella misurazione della lunghezza per la stringa «Düsseldorf-BVV-021», che contiene un carattere Unicode (Ü). L'`$strLenCP`operatore conta correttamente il numero di caratteri Unicode, mentre conta il `$strLenBytes` numero di byte.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$strLenCP` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

Node.js

Ecco un esempio di utilizzo dell'`$strLenCP`operatore in un' Node.js applicazione con il driver MongoDB:

```
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

Ecco un esempio di utilizzo dell'`$strLenCP`operatore in un'applicazione Python con il PyMongo driver:

```
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

L'`$strcasecmp`operatore in Amazon DocumentDB esegue un confronto tra due stringhe senza distinzione tra maiuscole e minuscole. Restituisce un valore intero che indica il confronto lessicografico delle due stringhe di input, ignorando le differenze tra maiuscole e minuscole.

Parametri

- `string1`: la prima stringa da confrontare.

- `string2`: La seconda stringa da confrontare.

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'`$strcasecmp`operatore per confrontare le stringhe di posizione della scrivania in una `people` raccolta, ignorando le differenze tra maiuscole e minuscole.

Crea documenti di esempio

```
db.people.insertMany([
  { "_id": 1, "Desk": "mke233-wi" },
  { "_id": 2, "Desk": "MKE233-WI" },
  { "_id": 3, "Desk": "mke233-wi" }
]);
```

Esempio di interrogazione

```
db.people.aggregate([
  {
    $project: {
      item: 1,
      compare: { $strcasecmp: ["$Desk", "mke233-wi"] }
    }
  }
]);
```

Output

```
{ "_id" : 1, "compare" : 0 }
{ "_id" : 2, "compare" : 0 }
{ "_id" : 3, "compare" : 0 }
```

L'output mostra che il confronto tra il `"Desk"` campo e la stringa viene `"mke233-wi"` restituito `0` per tutti e tre i documenti, a indicare che le stringhe sono uguali quando si ignorano le maiuscole e minuscole.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$strcasecmp` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$substr`operatore in Amazon DocumentDB viene utilizzato per estrarre una sottostringa da una determinata stringa. È particolarmente utile quando è necessario definire sottostringhe basate su un intervallo di caratteri anziché su un intervallo di byte. Ciò è particolarmente importante quando si ha a che fare con stringhe Unicode, in cui il numero di byte usati per rappresentare un carattere può variare.

Parametri

- `string`: La stringa di input da cui estrarre la sottostringa.
- `start`: la posizione iniziale (a base zero) della sottostringa da estrarre. Può essere un'espressione intera non negativa.
- `length`: il numero di caratteri nella sottostringa estratta. Può essere un'espressione intera non negativa.

Esempio (MongoDB Shell)

In questo esempio, dimostreremo l'uso di `$substr` per estrarre l'abbreviazione dello stato dalla posizione della scrivania di un dipendente.

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.people.aggregate([
  {
    $project: {
      "state": { $substr: ["$Desk", 12, 3] }
    }
  }
])
```

Output

```
{ "_id": 1, "state": "NRW" },
{ "_id": 2, "state": "HBB" },
{ "_id": 3, "state": "SHH" },
{ "_id": 4, "state": "BBB" }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$substr` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$substrBytes`operatore in Amazon DocumentDB viene utilizzato per estrarre una sottostringa da una stringa in base a un intervallo di byte specificato. Questo operatore è utile quando è necessario estrarre una sottostringa da una stringa e il numero di byte necessari per rappresentare ogni carattere della stringa è importante.

Al contrario`$substrCP`, che opera sul numero di punti di codice Unicode, `$substrBytes` opera sul numero di byte necessari per rappresentare i caratteri della stringa. Ciò può essere particolarmente utile quando si lavora con stringhe che contengono caratteri non ASCII, poiché la rappresentazione di tali caratteri può richiedere più di un byte.

*Nota: * `$substr` è obsoleto dalla versione 3.4. `$substr`è ora un alias per. `$substrBytes`

Parametri

- `string`: La stringa di input da cui verrà estratta la sottostringa.
- `startByte`: la posizione in byte iniziale a base zero della sottostringa da estrarre. È possibile utilizzare un valore negativo per specificare una posizione a partire dalla fine della stringa.
- `length`: il numero di byte nella sottostringa da estrarre.

Esempio (MongoDB Shell)

In questo esempio, useremo `$substrBytes` per estrarre una sottostringa da una stringa che contiene caratteri non ASCII.

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.people.aggregate([
  {
    $project: {
      "state": { $substrBytes: [ "$Desk", 12, 3] }
    }
  }
])
```

Output

```
{ "_id": 1, "state": "NRW" },
{ "_id": 2, "state": "HBB" },
{ "_id": 3, "state": "SHH" },
{ "_id": 4, "state": "BBB" }
```

In questo esempio, utilizziamo `$substrBytes` per estrarre una sottostringa di 3 byte a partire dal dodicesimo byte del campo. `Desk` Questo ci consente di estrarre l'abbreviazione di stato a 2 caratteri, anche se la stringa può contenere caratteri non ASCII.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$substrBytes` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$substrCP`operatore in Amazon DocumentDB viene utilizzato per estrarre una sottostringa da una stringa, in cui la sottostringa viene specificata come intervallo di punti di UTF-8 codice (CP). Questo operatore è particolarmente utile quando si lavora con stringhe Unicode, in quanto consente di estrarre sottostringhe senza doversi preoccupare della rappresentazione in byte sottostante dei caratteri.

A differenza dell'`$substrBytes`operatore, che opera sulle posizioni dei byte, l'`$substrCP`operatore lavora con le posizioni dei punti di codice. Ciò semplifica l'utilizzo di stringhe che contengono caratteri non ASCII, poiché il numero di punti di codice potrebbe non corrispondere al numero di byte o caratteri.

Parametri

- `string`: La stringa di input da cui estrarre la sottostringa.
- `start`: La posizione del punto di codice iniziale (a base zero) da cui estrarre la sottostringa.
- `length`: il numero di punti di codice da estrarre.

Esempio (MongoDB Shell)

In questo esempio, useremo l'`$substrCP`operatore per estrarre l'abbreviazione dello stato da una stringa contenente la posizione della scrivania del dipendente.

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.people.aggregate([
  {
    $project: {
      "state": { $substrCP: ["$Desk", 25, 2] }
    }
  }
]);
```

Output

```
{ "_id" : 1, "state" : "MN" }
{ "_id" : 2, "state" : "LA" }
{ "_id" : 3, "state" : "CA" }
```

In questo esempio, sappiamo che l'abbreviazione dello stato inizia dal 25° punto di codice del `Desk` campo e ha una lunghezza di 2 punti di codice. Utilizzando l'`$substrCP`operatore, possiamo estrarre l'abbreviazione dello stato senza doverci preoccupare della rappresentazione in byte sottostante della stringa.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$substrCP` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

In entrambi gli esempi Node.js e Python, utilizziamo l'`$substrCP`operatore per estrarre l'abbreviazione dello stato dal `Desk` campo, in modo simile all'esempio di MongoDB Shell.

$subtract

L'`$subtract`operatore in Amazon DocumentDB viene utilizzato per sottrarre valori. Può essere usato per sottrarre date, numeri o una combinazione di entrambi. Questo operatore è utile per calcolare la differenza tra due date o per sottrarre un valore da un numero.

Parametri

- `expression1`: il primo valore da sottrarre.
- `expression2`: Il secondo valore da cui sottrarre. `<expression1>`

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$subtract`operatore per calcolare la differenza tra due date.

Crea un documento di esempio

```
db.dates.insert([
  {
  "_id": 1,
  "startDate": ISODate("2023-01-01T00:00:00Z"),
  "endDate": ISODate("2023-01-05T12:00:00Z")
  }
]);
```

Esempio di query

```
db.dates.aggregate([
  {
    $project: {
      _id: 1,
      durationDays: {
        $divide: [
          { $subtract: ["$endDate", "$startDate"] },
          1000 * 60 * 60 * 24  // milliseconds in a day
        ]
      }
    }
  }
]);
```

Output

```
[ { _id: 1, durationDays: 4.5 } ]
```

In questo esempio, l'`$subtract`operatore viene utilizzato per calcolare la differenza tra i giorni `$endDate` e `$startDate` in giorni.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$subtract` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'$sumoperatore in Amazon DocumentDB restituisce la somma dell'espressione specificata per ogni documento in un gruppo. È un operatore di accumulazione di gruppo che viene in genere utilizzato nella fase $group di una pipeline di aggregazione per eseguire calcoli di sommatoria.

Parametri

- expression: L'espressione numerica da sommare. Può essere un percorso di campo, un'espressione o una costante.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'uso dell'$sumoperatore per calcolare le vendite totali per ciascun prodotto.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.sales.aggregate([
  { $group: {
      _id: "$product",
      totalSales: { $sum: { $multiply: [ "$price", "$quantity" ] } }
    }}
]);
```

Output

```
[
  { "_id": "abc", "totalSales": 50 },
  { "_id": "xyz", "totalSales": 120 }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$sum`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce la somma dell'espressione specificata per i documenti in ogni finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

Create documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`product`, ordina ogni partizione in base `day` e restituisce un totale progressivo `amount` compreso tra l'inizio della partizione e il documento corrente.

```
db.dailySales.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$product",
      sortBy: { day: 1 },
      output: {
        runningTotal: {
          $sum: "$amount",
          window: { documents: ["unbounded", "current"] }
        }
```

```
      }
    }
  }
]);
```

Output

```
[
  { "_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 }
]
```

Ogni documento è `runningTotal` arricchito dalla somma cumulativa contenuta `amount` nella sua partizione fino al documento corrente incluso.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'`$sum`operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano sia l'utilizzo dell'accumulatore (in`$group`) che l'utilizzo dell'operatore di finestra (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

Novità dalla versione 4.0.

Non supportato dal cluster Elastic.

L'`$switch`operatore è un operatore di espressione condizionale in Amazon DocumentDB che consente di valutare un elenco di espressioni di casi e restituire il valore del primo caso che restituisce true o un valore predefinito se nessuna espressione di case è vera.

Parametri

- `branches`: Una matrice di documenti, ognuno dei quali contiene un campo case che contiene l'espressione booleana da valutare e un campo then che contiene il valore da restituire se l'espressione case è vera.
- `default`: (opzionale) Il valore da restituire se nessuna delle espressioni maiuscole è vera.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$switch`operatore per determinare il costo di spedizione di un ordine in base al totale dell'ordine.

Crea documenti di esempio

```
db.orders.insertMany([
  { _id: 1, total: 50 },
  { _id: 2, total: 150 },
  { _id: 3, total: 250 }
]);
```

Esempio di query

```
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
        }
      }
    }
  }
])
```

Output

```
[
  {
    "_id": 1,
    "total": 50,
    "shippingCost": 5
  },
  {
```

```
    "_id": 2,
    "total": 150,
    "shippingCost": 10
  },
  {
    "_id": 3,
    "total": 250,
    "shippingCost": 15
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$switch` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

Nuovo dalla versione 8.0.1.

L'`$tan`operatore in Amazon DocumentDB restituisce la tangente di un valore misurato in radianti.

L'espressione di input deve essere convertita in un valore numerico. Se il valore è espresso in gradi, applicalo `$degreesToRadians` prima di passarlo a`$tan`.

Il tipo di ritorno è `double` predefinito. Se l'input è un decimale a 128 bit, anche l'output è un decimale a 128 bit.

Parametri

- `expression`: Un'espressione che si risolve in un numero in radianti.

Comportamento

null, NaN e +/- Infinity

Esempio	Risultati
`{ $tan: NaN }`	`NaN`
`{ $tan: null }`	`null`
`{ $tan: Infinity }` o `{ $tan: -Infinity }`	Genera un errore.

Quando l'input è `null` o il campo di riferimento è mancante, `$tan` restituisce. `null` Un input di `NaN` produce`NaN`. L'infinito positivo o negativo causa un errore perché la tangente non è definita a quei valori.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$tan`operatore per calcolare la tangente di un angolo in radianti.

Create documenti di esempio

```
db.angles.insertMany([
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.7853981633974483 },
  { "_id": 3, "angle": -0.7853981633974483 }
]);
```

Esempio di interrogazione

```
db.angles.aggregate([
  { $project: {
    "tangent": { $tan: "$angle" }
  }}
]);
```

Output

```
[
  { "_id": 1, "tangent": 0 },
  { "_id": 2, "tangent": 0.9999999999999999 },
  { "_id": 3, "tangent": -0.9999999999999999 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$tan` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

Nuovo dalla versione 8.0.1.

L'`$tanh`operatore in Amazon DocumentDB restituisce la tangente iperbolica di un valore.

L'espressione di input deve essere convertita in un valore numerico.

Il tipo restituito è `double` predefinito. Se l'input è un decimale a 128 bit, anche l'output è un decimale a 128 bit.

Parametri

- `expression`: Un'espressione che si risolve in un numero.

Comportamento

null, NaN e +/- Infinity

Esempio	Risultati
`{ $tanh: NaN }`	`NaN`
`{ $tanh: null }`	`null`
`{ $tanh: -Infinity }`	`-1`
`{ $tanh: Infinity }`	`1`

Quando l'input è `null` o il campo di riferimento è mancante, restituisce. `$tanh` `null` Un input di `NaN` produce`NaN`. Per l'infinito positivo il risultato è`1`; per l'infinito negativo il risultato è. `-1`

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$tanh`operatore per calcolare la tangente iperbolica di un valore.

Crea documenti di esempio

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Esempio di interrogazione

```
db.values.aggregate([
```

```
  { $project: {
    "result": { $tanh: "$value" }
  }}
]);
```

Output

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 0.7615941559557649 },
  { "_id": 3, "result": -0.7615941559557649 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$tanh` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

Nuovo dalla versione 8.0.1.

Usa l'`$top`accumulatore nella `$group` fase per restituire il documento con il punteggio più alto per gruppo in base a un ordinamento specificato.

Parametri

- `sortBy`: Un documento che specifica il criterio di ordinamento. `1`Da utilizzare in ordine crescente o `-1` decrescente.
- `output`: un'espressione che specifica i campi da restituire dal documento principale.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$top`accumulatore per trovare la vendita più venduta (quantità massima) per articolo in una raccolta di vendite.

Crea documenti di esempio

```
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 }
])
```

Esempio di interrogazione

```
db.sales.aggregate([
  { $group: { _id: "$item", topSale: { $top: { sortBy: { quantity: -1 }, output:
 { quantity: "$quantity", price: "$price" } } } } }
])
```

Output

```
[
  { "_id": "xyz", "topSale": { "quantity": 15, "price": 3 } },
  { "_id": "abc", "topSale": { "quantity": 10, "price": 5 } }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$top`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce il valore `output` del documento con il punteggio più alto (in base a quello dell'operatore`sortBy`) tra i documenti in ciascuna finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

> **Note**
>
> Se utilizzato come operatore di finestra in`$setWindowFields`, `$top` è limitato a 100 MB di dati intermedi. Un'operazione che supera questo limite restituisce un errore.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`player`, ordina ogni partizione in base `round` e restituisce il round con il punteggio più alto visto dall'inizio della partizione fino al documento corrente.

```
db.matchScores.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$player",
      sortBy: { round: 1 },
      output: {
        bestRoundSoFar: {
          $top: { sortBy: { score: -1 }, output: { round: "$round", score:
 "$score" } },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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 } }
```

```
]
```

Ogni documento è arricchito con `bestRoundSoFar` il punteggio più alto all'interno della sua partizione fino al documento corrente incluso.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'`$top`operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano sia l'utilizzo dell'accumulatore (in`$group`) che l'utilizzo dell'operatore di finestra (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()
```

$ a PN

Nuovo dalla versione 8.0.1.

Usa l'`$topN`accumulatore nella `$group` fase per restituire i primi N elementi all'interno di un gruppo in base a un ordinamento specificato. Se un gruppo contiene meno di N elementi, `$topN` restituisce tutti gli elementi del gruppo.

Parametri

- `n`: un numero intero positivo o un'espressione che si risolve in uno, che specifica quanti risultati principali restituire per gruppo.
- `sortBy`: un documento che specifica il criterio di ordinamento. `1`Da utilizzare in ordine crescente o `-1` decrescente.
- `output`: un'espressione che specifica i campi da restituire da ciascuno dei primi N documenti.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$topN`accumulatore per trovare le prime 2 vendite (quantità massima) per articolo in una raccolta di vendite.

Crea documenti di esempio

```
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 }
```

```
])
```

Esempio di interrogazione

```
db.sales.aggregate([
  { $group: { _id: "$item", topTwoSales: { $topN: { n: 2, sortBy: { quantity: -1 },
 output: { quantity: "$quantity", price: "$price" } } } } }
])
```

Output

```
[
  { "_id": "xyz", "topTwoSales": [{ "quantity": 15, "price": 3 }, { "quantity": 9,
 "price": 6 }] },
  { "_id": "abc", "topTwoSales": [{ "quantity": 10, "price": 5 }, { "quantity": 7,
 "price": 8 }] }
]
```

Esempio di utilizzo dell'operatore di finestra (MongoDB Shell)

Nuovo dalla versione 8.0.2.

L'`$topN`operatore può essere utilizzato anche come operatore di finestra nello `$setWindowFields` stage. In questo contesto, restituisce un array dei `output` primi N documenti (in base a quello dell'operatore`sortBy`) tra i documenti in ogni finestra. Si specifica l'operatore sotto il `output` campo e, facoltativamente, si definiscono i confini della finestra con un `window` documento.

> **Note**
>
> Se utilizzato come operatore di finestra in`$setWindowFields`, `$topN` è limitato a 100 MB di dati intermedi. Un'operazione che supera questo limite restituisce un errore.

Crea documenti di esempio

```
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 }
```

```
]);
```

Esempio di interrogazione

L'esempio seguente suddivide i documenti in base`player`, ordina ogni partizione in base `round` e restituisce i due round con il punteggio più alto visualizzati dall'inizio della partizione fino al documento corrente.

```
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"] }
        }
      }
    }
  }
]);
```

Output

```
[
  { "_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 }] }
]
```

Ogni documento è arricchito dai due round con `topTwoRoundsSoFar` il punteggio più alto all'interno della relativa partizione fino al documento corrente incluso. Se la finestra contiene meno di due documenti, vengono restituiti tutti.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$topN` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare. Gli esempi seguenti mostrano sia l'utilizzo dell'accumulatore (in`$group`) che l'utilizzo dell'operatore di finestra (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

Nuovo dalla versione 8.0.1.

L'`$trunc`operatore in Amazon DocumentDB tronca un numero a una cifra decimale specificata, rimuovendo le cifre senza arrotondare.

Parametri

- `number`: Un'espressione che si trasforma in un numero.
- `place`: facoltativo. Un'espressione intera compresa tra -20 e 100 che specifica il numero di cifre decimali da troncare. Un valore negativo viene troncato a sinistra del decimale. Il valore predefinito è 0.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$trunc`operatore per troncare i valori numerici a una cifra decimale.

Troncatura con un valore posizionale positivo

```
db.measurements.insertMany([
  {_id: 1, value: 3.456},
  {_id: 2, value: 7.891},
  {_id: 3, value: 12.345}
]);
```

Esempio di interrogazione

```
db.measurements.aggregate([
  { $project: { truncated: { $trunc: ["$value", 1] } } }
```

```
]);
```

Output

```
[
  {_id: 1, truncated: 3.4},
  {_id: 2, truncated: 7.8},
  {_id: 3, truncated: 12.3}
]
```

Troncatura con un valore posizionale negativo

Quando `place` è negativo, `$trunc` sostituisce le cifre a sinistra del decimale con zeri. L'esempio seguente tronca i valori utilizzando la prima cifra a sinistra del decimale.

```
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] } } }
]);
```

Output

```
[
  {_id: 1, truncated: 10},
  {_id: 2, truncated: 20},
  {_id: 3, truncated: 30}
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'`$trunc`operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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()
```

Incremento $TS

Nuovo dalla versione 8.0.1.

L'`$tsIncrement`operatore in Amazon DocumentDB restituisce l'ordinale incrementale da un valore di Timestamp come numero intero lungo.

Parametri

- `expression`: Un'espressione che si risolve in un Timestamp.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$tsIncrement`operatore per estrarre l'incremento ordinale dai valori di Timestamp.

Crea documenti di esempio

```
db.events.insertMany([
  {_id: 1, ts: Timestamp(1678900000, 1)},
  {_id: 2, ts: Timestamp(1678900000, 2)},
  {_id: 3, ts: Timestamp(1678900001, 1)}
]);
```

Esempio di interrogazione

```
db.events.aggregate([
  { $project: { increment: { $tsIncrement: "$ts" } } }
]);
```

Output

```
[
  {_id: 1, increment: Long("1")},
  {_id: 2, increment: Long("2")},
  {_id: 3, increment: Long("1")}
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$tsIncrement` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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()
```

$tsSecondo

Nuovo dalla versione 8.0.1.

L'`$tsSecond`operatore in Amazon DocumentDB restituisce la parte in secondi di un valore di Timestamp come numero intero lungo (i secondi dell'epoca Unix).

Parametri

- `expression`: Un'espressione che si risolve in un Timestamp.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$tsSecond`operatore per estrarre la parte dei secondi dai valori di Timestamp.

Crea documenti di esempio

```
db.events.insertMany([
  {_id: 1, ts: Timestamp(1678900000, 1)},
  {_id: 2, ts: Timestamp(1678900000, 2)},
  {_id: 3, ts: Timestamp(1678900001, 1)}
]);
```

Esempio di interrogazione

```
db.events.aggregate([
  { $project: { seconds: { $tsSecond: "$ts" } } }
]);
```

Output

```
[
  {_id: 1, seconds: Long("1678900000")},
  {_id: 2, seconds: Long("1678900000")},
  {_id: 3, seconds: Long("1678900001")}
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$tsSecond` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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()
```

$ a BOOL

L'`$toBool`operatore in Amazon DocumentDB converte un'espressione in un valore booleano.

Parametri

- `expression`: L'espressione da convertire in un valore booleano.

Nota: qualsiasi stringa viene convertita in. `true`

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo `$toBool` dell'operatore per normalizzare i valori dello stato del dispositivo da diversi tipi di dati.

Crea documenti di esempio

```
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 }
]);
```

Esempio di query

```
db.deviceStates.aggregate([
  {
    $project: {
      _id: 1,
      deviceId: 1,
      isActive: { $toBool: "$status" }
    }
  }
]);
```

Output

```
[
  { "_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 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$toBool` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$ fino ad oggi

Nuovo dalla versione 4.0

L'operatore di `$toDate` aggregazione in Amazon DocumentDB viene utilizzato per convertire una stringa di data o data e ora in un tipo di data BSON. Questa è l'operazione inversa dell'operatore. `$dateToString`

Parametri

- `dateString`: Una rappresentazione in formato stringa di una data o di una data e ora da convertire in un tipo di data BSON.
- `format`: (opzionale) Una stringa che specifica il formato di. `dateString` Se non viene fornito, l'operatore tenterà di analizzarli `dateString` in vari formati standard di data e ora.
- `timezone`: (opzionale) Una stringa che rappresenta il fuso orario da utilizzare per la conversione. Se non viene fornito, viene utilizzato il fuso orario locale.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$toDate`operatore per convertire una stringa di data in un tipo di data BSON.

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.events.aggregate([
  {
    $project: {
      eventName: 1,
      eventTimeDate: { $toDate: "$eventTime" }
    }
  }
]);
```

Output

```
[
  {
    "_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")
  }
```

```
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$toDate` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$ a Decimal

Nuovo dalla versione 4.0

L'`$toDecimal`operatore in Amazon DocumentDB viene utilizzato per convertire un valore nel tipo di dati Decimal128. Ciò è utile quando è necessario eseguire un'aritmetica decimale precisa o gestire valori decimali di grandi dimensioni che non possono essere rappresentati con precisione utilizzando il tipo di dati Double.

Parametri

- `expression`: espressione da convertire nel tipo di dati Decimal128.

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'`$toDecimal`operatore per convertire un valore di stringa in un tipo di dati Decimal128.

Crea documenti di esempio

```
db.numbers.insertOne({ _id: 1, value: "3.14" });
db.numbers.insertOne({ _id: 2, value: "2.71" });
```

Esempio di interrogazione

```
db.numbers.aggregate([
```

```
  { $project: {
    _id: 1,
    decimalValue: { $toDecimal: "$value" }
  }}
])
```

Output

```
[
  { "_id" : 1, "decimalValue" : Decimal128("3.14") },
  { "_id" : 2, "decimalValue" : Decimal128("2.71") }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$toDecimal` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$ a Double

Nuovo dalla versione 4.0

L'`$toDouble`operatore in Amazon DocumentDB viene utilizzato per convertire un valore in un numero a virgola mobile a 64 bit a precisione doppia. Questo può essere utile quando è necessario eseguire operazioni aritmetiche su valori che originariamente non erano in formato numerico.

Parametri

`<expression>`: L'espressione da convertire in un valore doppio. Può essere qualsiasi espressione valida che si risolve in un valore numerico, di stringa o booleano.

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'`$toDouble`operatore per convertire un valore di stringa in un valore numerico allo scopo di eseguire un calcolo matematico.

Crea documenti di esempio

```
db.numbers.insertMany([
  { _id: 1, value: "10.5" },
  { _id: 2, value: "20.25" },
  { _id: 3, value: "7" }
])
```

Esempio di interrogazione

```
db.numbers.aggregate([
  {
    $project: {
      _id: 1,
      value: 1,
      double_value: { $toDouble: "$value" },
      double_plus_five: { $add: [{ $toDouble: "$value" }, 5] }
    }
  }
])
```

Output

```
[
  { "_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 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$toDouble` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$ a INT

Nuovo dalla versione 4.0

L'`$toInt`operatore in Amazon DocumentDB viene utilizzato per convertire un valore di input in un tipo di dati intero. Questo operatore è utile quando è necessario garantire che un campo o un'espressione sia rappresentato come un numero intero, il che può essere importante per determinate operazioni o attività di elaborazione dei dati.

Parametri

- `expression`: L'espressione da convertire in un numero intero.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$toInt`operatore per convertire un valore di stringa in un numero intero.

Crea documenti di esempio

```
db.numbers.insertMany([
  { "name": "one", "value": "1" },
  { "name": "hundred", "value": "100" }
]);
```

Esempio di interrogazione

```
db.numbers.aggregate([
  { $project: {
    "_id": 0,
    "name": 1,
    "intValue": { $toInt: "$value" }
  }}
]);
```

Output

```
{ "name": "one", "intValue": 1 }
{ "name": "hundred", "intValue": 100 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$toInt` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$ a Long

Novità dalla versione 4.0

L'`$toLong`operatore in Amazon DocumentDB viene utilizzato per convertire un valore in un tipo di dati intero (lungo) a 64 bit. Questo può essere utile quando è necessario eseguire operazioni aritmetiche o confronti su valori numerici che possono essere memorizzati come stringhe o altri tipi di dati.

Parametri

- `expression`: L'espressione da convertire in un numero intero a 64 bit.

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'`$toLong`operatore per convertire un valore di stringa in un numero intero a 64 bit.

Crea documenti di esempio

```
db.numbers.insertMany([
  { _id: 1, value: "42" },
  { _id: 3, value: "92233720368547758O7" }
]);
```

Esempio di interrogazione

```
db.numbers.aggregate([
  {
    $project: {
      _id: 1,
      longValue: { $toLong: "$value" }
    }
  }
])
```

Output

```
[
  { "_id" : 1, "longValue" : 42 },
  { "_id" : 3, "longValue" : 9223372036854775807 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$toLong` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$toLower`operatore in Amazon DocumentDB viene utilizzato per convertire una stringa in lettere minuscole.

Parametri

- `expression`: L'espressione stringa da convertire in lettere minuscole.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$toLower`operatore per convertire il `Desk` campo in lettere minuscole.

Crea documenti di esempio

```
db.locations.insertMany([
  { "_id": 1, "Desk": "Düsseldorf-BVV-021" },
  { "_id": 2, "Desk": "Munich-HGG-32a" }
]);
```

Esempio di interrogazione

```
db.locations.aggregate([
  { $project: { item: { $toLower: "$Desk" } } }
]);
```

Output

```
{ "_id" : 1, "item" : "düsseldorf-bvv-021" }
{ "_id" : 2, "item" : "munich-hgg-32a" }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$toLower` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$a ObjectId

Nuovo dalla versione 4.0

L'`$toObjectId`operatore in Amazon DocumentDB viene utilizzato per convertire una rappresentazione in formato stringa di un tipo di dati ObjectId in un tipo di ObjectId dati effettivo. Ciò può essere utile quando si lavora con dati che sono stati archiviati come rappresentazioni di stringhe di ObjectIds, in quanto consente di eseguire operazioni che richiedono il tipo di ObjectId dati.

Parametri

- `expression`: Un'espressione stringa che rappresenta un valore valido ObjectId.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$toObjectId`operatore per convertire una rappresentazione in formato stringa di an nel ObjectId tipo di ObjectId dati.

Crea documenti di esempio

```
db.employees.insertMany([
  { _id: 1, empId:"64e5f8886218c620cf0e8f8a", name: "Carol Smith", employeeId:
 "c720a" },
  { _id: 2, empId:"64e5f94e6218c620cf0e8f8c", name: "Bill Taylor", employeeId:
 "c721a" }
]);
```

Esempio di interrogazione

```
db.employees.aggregate([
  { $project: {
    "empIdAsObjectId": {$toObjectId: "$empId"}}
  }
]);
```

Output

```
[
  { _id: 1, empIdAsObjectId: ObjectId('64e5f8886218c620cf0e8f8a') },
  { _id: 2, empIdAsObjectId: ObjectId('64e5f94e6218c620cf0e8f8c') }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$toObjectId` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

Nuovo dalla versione 4.0

L'`$toString`operatore in Amazon DocumentDB viene utilizzato per convertire un valore di qualsiasi tipo (ad eccezione di null) in una rappresentazione di stringa. Questo può essere utile quando è necessario eseguire operazioni sulle stringhe su valori che originariamente non erano in formato stringa.

Parametri

- `expression`: L'espressione da convertire in una stringa.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$toString`operatore per convertire valori numerici in stringhe.

Crea documenti di esempio

```
db.numbers.insertMany([
  { "_id": 1, "value": 42 },
  { "_id": 2, "value": 3.14 }
]);
```

Esempio di interrogazione

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    valueAsString: { $toString: "$value" }
  }}
]);
```

Output

```
{ "_id": 1, "valueAsString": "42" }
{ "_id": 2, "valueAsString": "3.14" }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$toString` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$toUpper`operatore in Amazon DocumentDB viene utilizzato per convertire una stringa in lettere maiuscole.

Parametri

- `expression`: L'espressione stringa da convertire in lettere maiuscole.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$toUpper`operatore per convertire il `Desk` campo in lettere maiuscole.

Crea documenti di esempio

```
db.locations.insertMany([
  { "_id": 1, "Desk": "düsseldorf-bvv-021" },
  { "_id": 2, "Desk": "munich-hgg-32a" }
]);
```

Esempio di query

```
db.locations.aggregate([
  { $project: { item: { $toUpper: "$Desk" } } }
]);
```

Output

```
{ "_id" : 1, "item" : "DüSSELDORF-BVV-021" }
{ "_id" : 2, "item" : "MUNICH-HGG-32A" }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$toUpper` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

da $ a UUID

Nuovo dalla versione 8.0.1.

L'`$toUUID`operatore in Amazon DocumentDB converte un valore di stringa in un UUID (sottotipo binario 4).

Parametri

- `expression`: Un'espressione che si risolve in una stringa in formato UUID (ad esempio, «12345678-1234-1234-1234-1234-1234-123456789abc»).

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$toUUID`operatore per convertire i valori delle stringhe in formato UUID.

Crea documenti di esempio

```
db.records.insertMany([
  {_id: 1, uuidStr: "550e8400-e29b-41d4-a716-446655440000"},
  {_id: 2, uuidStr: "6ba7b810-9dad-11d1-80b4-00c04fd430c8"}
]);
```

Esempio di interrogazione

```
db.records.aggregate([
  { $project: { uuid: { $toUUID: "$uuidStr" } } }
]);
```

Output

```
[
  {_id: 1, uuid: UUID("550e8400-e29b-41d4-a716-446655440000")},
  {_id: 2, uuid: UUID("6ba7b810-9dad-11d1-80b4-00c04fd430c8")}
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$toUUID` dell'operatore, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

Nuovo dalla versione 4.0

L'`$trim`operatore in Amazon DocumentDB viene utilizzato per rimuovere gli spazi bianchi and/or iniziali da una stringa.

Parametri

- `input`: L'espressione della stringa da tagliare.
- `chars`: (opzionale) Specifica i caratteri da tagliare dall'inizio e dalla fine dell'input, l'impostazione predefinita è lo spazio bianco.

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare l'`$trim`operatore per rimuovere gli spazi bianchi iniziali e finali da una stringa.

Crea documenti di esempio

```
db.people.insertMany([
  { "name": "   John Doe   " },
  { "name": "   Bob Johnson   " }
])
```

Esempio di interrogazione

```
db.people.aggregate([
  { $project: {
    "name": { $trim: {input: "$name"}}
  }}
])
```

Output

```
[
  { "name": "John Doe" },
  { "name": "Bob Johnson" }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$trim` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'operatore di `$type` aggregazione restituisce il tipo di dati BSON di un campo specificato. Ciò è utile per identificare il tipo di dati dei valori dei campi durante le operazioni di aggregazione.

Parametri

- `expression`: il campo o l'espressione di cui restituire il tipo.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo `$type` dell'operatore per identificare il tipo di dati del campo del prezzo per ciascun prodotto.

Crea documenti di esempio

```
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 }
]);
```

Esempio di query

```
db.inventory.aggregate([
  {
    $project: {
      item: 1,
      price: 1,
      priceType: { $type: "$price" }
    }
  }
```

```
]);
```

Output

```
[
  { _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' }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di `$type` aggregazione, scegli la scheda relativa alla lingua che desideri utilizzare:

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

Novità dalla versione 8.0

Non supportato dal cluster Elastic.

La fase `$unset` di aggregazione in Amazon DocumentDB consente di rimuovere campi dai documenti.

Parametri

- `expression`: nome del campo o elenco di più nomi di campo.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'uso della fase di `$unset` aggregazione per rimuovere il campo. `price`

Crea documenti di esempio

```
db.inventory.insertMany([
  { item: "pencil", quantity: 100, price: 0.24},
  { item: "pen", quantity: 204, price: 1.78 }
]);
```

Esempio di aggregazione

```
db.inventory.aggregate([
  {
    $unset: "price"
  }
])
```

Output

```
[
  {
    _id: ObjectId('69248951d66dcae121d2950d'),
    item: 'pencil',
    quantity: 100
  },
  {
    _id: ObjectId('69248951d66dcae121d2950e'),
    item: 'pen',
    quantity: 204
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$unset` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$unwind`operatore viene utilizzato per decostruire un campo array dai documenti di input per generare un documento per ogni elemento. Ciò può essere utile quando si desidera eseguire operazioni sui singoli elementi di un array, come filtrare, ordinare o trasformare i dati.

Parametri

- `path`: il percorso del campo dell'array da srotolare.
- `includeArrayIndex`: (opzionale) Specificate il nome del nuovo campo che contiene l'indice dell'elemento dell'array.
- `preserveNullAndEmptyArrays`: (opzionale) Determina se l'operazione mantiene il documento originale quando il campo dell'array è nullo o è un array vuoto.

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare l'`$unwind`operatore per decostruire un campo array ed eseguire ulteriori operazioni sui singoli elementi.

Crea documenti di esempio

```
db.people.insertMany([
  { _id: 1, name: "jon", hobbies: ["painting", "dancing", "singing"] },
  { _id: 2, name: "jane", hobbies: ["reading", "swimming"] },
  { _id: 3, name: "jack", hobbies: [] }
])
```

Esempio di query

```
db.people.aggregate([
  { $unwind: "$hobbies" }
])
```

Output

```
[
```

```
  { _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' }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$unwind` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$ Ricerca vettoriale

Nuovo dalla versione 8.0

Non supportato dal cluster Elastic.

L'`$vectorSearch`operatore in Amazon DocumentDB consente di eseguire ricerche vettoriali, un metodo utilizzato nell'apprendimento automatico per trovare punti dati simili confrontando le loro rappresentazioni vettoriali utilizzando metriche di distanza o similarità. Questa funzionalità combina la flessibilità e l'ampia gamma di interrogazioni di un database di JSON-based documenti con la potenza della ricerca vettoriale, consentendoti di creare casi d'uso di machine learning e intelligenza artificiale generativa come la ricerca semantica, la raccomandazione di prodotti e altro ancora.

Parametri

- `<exact>`(opzionale): contrassegno che specifica se eseguire la ricerca Exact Nearest Neighbor (ENN) o Approximate Nearest Neighbor (ANN). Il valore può essere uno dei seguenti:
- false: per eseguire una ricerca ANN
- true - per eseguire una ricerca ENN

Se omesso o impostato su false, `numCandidates` è obbligatorio.

```
- `<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.
```

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$vectorSearch`operatore per trovare descrizioni di prodotti simili in base alle loro rappresentazioni vettoriali.

Crea documenti di esempio

```
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]
  }
]);
```

Crea un indice di ricerca vettoriale

```
db.runCommand(
    {
        createIndexes: "products",
        indexes: [{
            key: {
                "description_vector": "vector"
            },
            vectorOptions: {
                type: "hnsw",
                dimensions: 3,
                similarity: "cosine",
```

```
                m: 16,
                efConstruction: 64
            },
            name: "description_index"
        }]
    }
);
```

Esempio di query

```
db.products.aggregate([
  { $vectorSearch: {
      index: "description_index",
      limit: 2,
      numCandidates: 10,
      path: "description_vector",
      queryVector: [0.1, 0.2, 0.3]
    }
  }
]);
```

Output

```
[
  {
    "_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 ]
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$vectorSearch` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'$weekoperatore in Amazon DocumentDB restituisce il numero della settimana di una data (0-53) in base allo standard ISO 8601. Il numero della settimana viene calcolato in base all'anno e al giorno della settimana, con lunedì come primo giorno della settimana.

Parametri

Nessuno

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'$weekoperatore per recuperare il numero della settimana di una determinata data.

Crea documenti di esempio

```
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") }
]);
```

Esempio di query

```
db.events.aggregate([
```

```
  { $project: {
    _id: 1,
    week: { $week: "$date" }
  }}
]);
```

Output

```
[
  { "_id": 1, "week": 1 },
  { "_id": 2, "week": 2 },
  { "_id": 3, "week": 53 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$week` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$year`operatore in Amazon DocumentDB estrae il componente anno da una data o un timestamp.

Parametri

- `expression`: L'espressione della data o dell'ora da cui estrarre il componente dell'anno.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$year`operatore per estrarre il componente anno da un campo data.

Crea documenti di esempio

```
db.events.insertMany([
  { "_id": 1, "date": ISODate("2023-04-15T00:00:00Z") },
```

```
  { "_id": 3, "date": ISODate("2021-12-31T00:00:00Z") }
]);
```

Esempio di interrogazione

```
db.events.aggregate([
  { $project: { year: { $year: "$date" } } }
]);
```

Output

```
[
  { "_id": 1, "year": 2023 },
  { "_id": 3, "year": 2021 }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$year` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$zip`operatore in Amazon DocumentDB consente di combinare più array in un unico array di tuple (coppie chiave-valore). Ciò è utile quando è necessario creare nuovi documenti o oggetti combinando dati provenienti da diverse fonti o matrici all'interno di un documento.

Parametri

- `inputs`: Una matrice di espressioni che si risolvono in matrici. Questi array verranno combinati in un unico array di tuple.
- `useLongestLength`: (opzionale) Se`true`, l'array di output avrà la lunghezza dell'array di input più lungo, riempirà di valori gli array più corti. `null` Se`false`, l'array di output avrà la lunghezza dell'array di input più corto.
- `defaults`: (opzionale) Un array di valori predefiniti da utilizzare per le tuple se l'array di input corrispondente è più corto dell'array di input più lungo e lo è. `useLongestLength` `true`

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$zip`operatore per combinare due array in un unico array di tuple.

Crea documenti di esempio

```
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"]
  }
])
```

Esempio di interrogazione

```
db.grades.aggregate([
  {
    $project: {
      "name": 1,
      "scoredClasses": {
        $zip: {
          inputs: ["$scores", "$classes"],
          useLongestLength: true,
          defaults: [null, null]
        }
      }
    }
  }
])
```

Output

```
[
  {
```

```
    "_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"]
    ]
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$zip` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

Dati geospaziali

Questa sezione fornisce informazioni dettagliate sugli operatori geospaziali supportati da Amazon DocumentDB.

Argomenti

- $geometry
- $geoIntersects
- $geoWithin
- $maxDistance
- $minDistance
- $near
- $nearSphere

$geometry

L'`$geometry`operatore in Amazon DocumentDB viene utilizzato per specificare un oggetto geometrico GeoJSON come parte di una query geospaziale. Questo operatore viene utilizzato insieme ad altri operatori di interrogazione geospaziali come e per eseguire query spaziali sui dati. `$geoWithin` `$geoIntersects`

In Amazon DocumentDB, l'`$geometry`operatore supporta i seguenti tipi di geometria GeoJSON:

- Point (Punto)
- LineString
- Poligono
- MultiPoint
- MultiLineString
- MultiPolygon
- GeometryCollection

Parametri

- `type`: Il tipo di oggetto geometrico GeoJSON, ad esempio, ecc. `Point` `Polygon`
- `coordinates`: Un array di coordinate che rappresentano la geometria. La struttura dell'array di coordinate dipende dal tipo di geometria.

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare l'`$geometry`operatore per eseguire una `$geoIntersects` query in Amazon DocumentDB.

Crea documenti di esempio

```
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]
      ]]
    }
  }
]);
```

Esempio di interrogazione

```
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]
        ]]
      }
    }
  }
})
```

Output

```
[
  {
    "_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]
        ]
      ]
    }
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$geometry` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$geoIntersects`operatore in Amazon DocumentDB viene utilizzato per trovare documenti i cui dati geospaziali si intersecano con un oggetto GeoJSON specificato. Questo operatore è utile per le applicazioni che richiedono l'identificazione dei documenti in base alla loro relazione spaziale con una determinata forma geografica, come un poligono o un multipoligono.

Parametri

- `$geometry`: Un oggetto GeoJSON che rappresenta la forma da verificare per l'intersezione. I tipi di oggetti GeoJSON supportati `Point` sono`LineString`,, `Polygon` e. `MultiPolygon`

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$geoIntersects`operatore per trovare il nome dello stato per un determinato set di coordinate in Amazon DocumentDB.

Crea documenti di esempio

```
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]
      ]]
    }
  }
]);
```

Esempio di interrogazione

```
var location = [-73.965355, 40.782865];

db.states.find({
  "loc": {
    "$geoIntersects": {
```

```
      "$geometry": {
        "type": "Point",
        "coordinates": location
      }
    }
  }
}, {
  "name": 1
});
```

Output

```
{ "_id" : ObjectId("536b0a143004b15885c91a2c"), "name" : "New York" }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$geoIntersects` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$geoWithin`operatore in Amazon DocumentDB viene utilizzato per trovare documenti i cui dati di posizione (rappresentati come oggetti GeoJSON) sono completamente contenuti all'interno di una forma specificata, come un poligono o un multipoligono. Ciò è utile per eseguire query su oggetti che si trovano all'interno di una regione geografica specifica.

Parametri

- `$geometry`: Un oggetto GeoJSON che rappresenta la forma su cui eseguire la query.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$geoWithin`operatore per trovare tutti gli aeroporti situati nello stato di New York.

Crea documenti di esempio

```
// 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]
        }
    }
]);
```

Esempio di interrogazione

```
var state = db.states.findOne({"name": "New York"});

db.airports.find({
    "loc": {
        "$geoWithin": {
```

```
            "$geometry": state.loc
        }
    }
}, {
    "name": 1,
    "type": 1,
    "code": 1,
    "_id": 0
});
```

Output

```
[
  {
    "name": "John F. Kennedy International Airport",
    "type": "airport",
    "code": "JFK"
  },
  {
    "name": "LaGuardia Airport",
    "type": "airport",
    "code": "LGA"
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$geoWithin` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$maxDistance`operatore in Amazon DocumentDB viene utilizzato per specificare la distanza massima (in metri) da un punto GeoJSON entro cui i documenti devono trovarsi per essere inclusi nei risultati della query. Questo operatore viene utilizzato insieme all'operatore per eseguire interrogazioni `$nearSphere` geospaziali.

Parametri

- `$maxDistance`: La distanza massima (in metri) dal punto di riferimento entro il quale i documenti devono trovarsi per essere inclusi nei risultati dell'interrogazione.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$maxDistance`operatore in Amazon DocumentDB per trovare tutte le capitali degli stati entro 100 chilometri da Boston.

Crea documenti di esempio

```
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] } }
]);
```

Esempio di interrogazione

```
db.capitals.find(
  {
    location: {
      $nearSphere: {
        $geometry: { type: "Point", coordinates: [-71.0589, 42.3601] },
        $maxDistance: 100000
      }
    }
  },
```

```
  { state: 1, city: 1, _id: 0 }
);
```

Output

```
[
  { "state": "Rhode Island", "city": "Providence" },
  { "state": "New Hampshire", "city": "Concord" }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$maxDistance` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

`$minDistance`è un operatore di ricerca utilizzato insieme `$nearSphere` o `$geoNear` per filtrare documenti che si trovano almeno alla distanza minima specificata dal punto centrale. Questo operatore è supportato in Amazon DocumentDB e funziona in modo simile alla sua controparte in MongoDB.

Parametri

- `$minDistance`: La distanza minima (in metri) dal punto centrale per includere i documenti nei risultati.

Esempio (MongoDB Shell)

In questo esempio, troveremo tutti i ristoranti entro un raggio di 2 chilometri da una località specifica a Seattle, Washington.

Crea documenti di esempio

```
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]
    }
  }
]);
```

Esempio di query

```
db.usarestaurants.find({
```

```
  "location": {
    "$nearSphere": {
      "$geometry": {
        "type": "Point",
        "coordinates": [-122.3516, 47.6156]
      },
      "$minDistance": 1,
      "$maxDistance": 2000
    }
  }
}, {
  "name": 1
});
```

Output

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
{ "_id" : ObjectId("611f3da985009a81ad38e74c"), "name" : "Pike Place Grill" }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$minDistance` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$near`operatore in Amazon DocumentDB viene utilizzato per trovare documenti geograficamente vicini a un punto specifico. Restituisce i documenti ordinati in base alla distanza, con i documenti più vicini per primi. Questo operatore richiede un indice geospaziale a due sfere ed è utile per le interrogazioni di prossimità sui dati di localizzazione.

Parametri

- `$geometry`: un oggetto GeoJSON Point che definisce il punto centrale per la query Near.
- `$maxDistance`: (opzionale) La distanza massima in metri dal punto specificato che un documento può raggiungere per corrispondere alla query.
- `$minDistance`: (opzionale) La distanza minima in metri dal punto specificato che un documento può avere per corrispondere alla query.

Requisiti dell'indice

- `2dsphere index`: necessario per le interrogazioni geospaziali sui dati GeoJSON Point.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$near`operatore per trovare i ristoranti più vicini a una località specifica a Seattle, Washington.

Crea documenti di esempio

```
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] }
  }
]);
```

Crea un indice a due sfere

```
db.usarestaurants.createIndex({ "location": "2dsphere" });
```

Esempio di query con GeoJSON Point

```
db.usarestaurants.find({
  location: {
    $near: {
      $geometry: {
        type: "Point",
        coordinates: [-122.3516, 47.6156]
      },
      $maxDistance: 100,
      $minDistance: 10
    }
  }
});
```

Output

```
{
  "_id" : ObjectId("69031ec9ea1c2922a1ce5f4a"),
  "name" : "Noodle House",
  "city" : "Seattle",
  "state" : "Washington",
  "rating" : 4.8,
  "location" : {
    "type" : "Point",
    "coordinates" : [ -122.3517, 47.6159 ]
  }
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$near` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$nearSphere`operatore in Amazon DocumentDB viene utilizzato per trovare documenti che si trovano entro una distanza specificata da un punto geospaziale. Questo operatore è particolarmente utile per le interrogazioni geospaziali, ad esempio per trovare tutti i ristoranti entro un certo raggio da una determinata località.

Parametri

- `$geometry`: un oggetto GeoJSON che rappresenta il punto di riferimento. Deve essere un `Point` oggetto con `type` e `coordinates` campi.

- `$minDistance`: (opzionale) La distanza minima (in metri) dal punto di riferimento a cui devono essere i documenti.
- `$maxDistance`: (opzionale) La distanza massima (in metri) dal punto di riferimento a cui devono essere i documenti.

Esempio (MongoDB Shell)

In questo esempio, troveremo tutti i ristoranti nel raggio di 2 chilometri (2000 metri) da una località specifica a Seattle, WA.

Crea documenti di esempio

```
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] }
  }
]);
```

Esempio di interrogazione

```
db.usarestaurants.find({
  location: {
    $nearSphere: {
      $geometry: {
        type: "Point",
        coordinates: [-122.3516, 47.6156]
      },
      $minDistance: 1,
      $maxDistance: 2000
    }
  }
}, {
```

```
  name: 1
});
```

Output

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
{ "_id" : ObjectId("611f3da985009a81ad38e74c"), "name" : "Pike Place Grill" }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$nearSphere` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

Operatori di interrogazione e proiezione

Questa sezione fornisce informazioni dettagliate sugli operatori di interrogazione e proiezione supportati da Amazon DocumentDB.

Argomenti

- $

- $all
- $and
- $ bit AllClear
- $ bit AllSet
- $ bit AnyClear
- $ bit AnySet
- $comment
- $elemMatch
- $eq
- $exists
- $expr
- $gt
- $gte
- $in
- $jsonSchema
- $lt
- $mod
- $lte
- $meta
- $ne
- $nin
- $nor
- $not
- $or
- $regex
- $Frequenza di campionamento
- $slice
- $size
- $text

- $type

$

L'operatore di $ proiezione limita il contenuto di un campo di matrice per restituire solo il primo elemento che corrisponde alla condizione di interrogazione. Viene utilizzato per proiettare un singolo elemento dell'array corrispondente.

Parametri

- `field.$`: Il campo dell'array con l'operatore posizionale per proiettare il primo elemento corrispondente.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'operatore di $ proiezione per restituire solo l'elemento dell'array corrispondente.

Create documenti di esempio

```
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] }
]);
```

Esempio di interrogazione

```
db.students.find(
  { grades: { $gte: 90 } },
  { name: 1, "grades.$": 1 }
);
```

Output

```
{ "_id" : 1, "name" : "Alice", "grades" : [ 92 ] }
{ "_id" : 2, "name" : "Bob", "grades" : [ 92 ] }
{ "_id" : 3, "name" : "Charlie", "grades" : [ 95 ] }
```

In questo esempio, per ogni studente viene restituito solo il primo voto maggiore o uguale a 90.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di $ proiezione, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$all`operatore in Amazon DocumentDB viene utilizzato per abbinare documenti in cui il valore di un campo è un array e contiene tutti gli elementi specificati, indipendentemente dall'ordine degli elementi nell'array.

Parametri

- `field`: il nome del campo da controllare.
- `[value1, value2, ...]`: l'elenco dei valori da abbinare nell'array.

Utilizzo **`$elemMatch`** all'interno di un'**`$all`**espressione

Utilizzo di $elemMatch all'interno di un'espressione $all Per le limitazioni relative all'uso dell'`$elemMatch`operatore all'interno di un'`$all`espressione, vedere.

Dollaro ($) nei nomi dei campi

Dollaro ($) e punto (.) nei nomi dei campiPer le limitazioni relative all'interrogazione di campi con $ prefisso `$all` negli oggetti annidati, vedere.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$all`operatore per recuperare documenti in cui il campo «Colors» è un array che contiene sia «Red» che «Blue».

Crea documenti di esempio

```
db.example.insertMany([
  { "Item": "Pen", "Colors": ["Red", "Blue", "Green"] },
  { "Item": "Notebook", "Colors": ["Blue", "White"] },
  { "Item": "Poster Paint", "Colors": ["Red", "Yellow", "White"] }
```

```
])
```

Esempio di interrogazione

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

Output

```
{
  "_id" : ObjectId("6137d6c5b3a1d35e0b6ee6ad"),
  "Item" : "Pen",
  "Colors" : [
          "Red",
          "Blue",
          "Green"
  ]
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$all` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$and`operatore in Amazon DocumentDB viene utilizzato per combinare più espressioni e valutarle come un'unica condizione. Restituisce `true` se tutte le espressioni fornite restituiscono lo stesso risultato`true`, e in `false` caso contrario. Questo operatore è utile per applicare più criteri a un'interrogazione.

Parametri

- `expression1`: un'espressione obbligatoria che restituisce un valore booleano.
- `expression2`: Un'espressione obbligatoria che restituisce un valore booleano.
- `...`: espressioni obbligatorie aggiuntive che restituiscono valori booleani.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'uso dell'`$and`operatore per trovare tutti i documenti nella raccolta «users» in cui il campo «age» è maggiore di 18 e il campo «status» è «attivo».

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.users.find({
  $and: [
    { age: { $gt: 18 } },
    { status: "active" }
  ]
});
```

Output

```
[
  { "_id" : ObjectId("614e3c4b63f5892e7c4e2345"), "name" : "John", "age" : 25,
 "status" : "active" },
  { "_id" : ObjectId("614e3c4b63f5892e7c4e2347"), "name" : "Alice", "age" : 22,
 "status" : "active" }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del $and comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

$ bit AllClear

L'`$bitsAllClear`operatore in Amazon DocumentDB viene utilizzato per abbinare i documenti in cui tutti i bit specificati in un campo sono cancellati (impostato su 0). Questo operatore può essere utile per eseguire operazioni bit per bit sui dati archiviati.

Parametri

- `field`: Il campo per verificare che i bit specificati vengano cancellati.

- `value`: La maschera di bit numerica che specifica quali bit devono essere controllati o un elenco di posizioni dei bit da controllare. Una maschera di bit numerica può essere in forma binaria (0b...), decimale, esadecimale (0x...), ottale (0o...) o binaria (). BinData In un elenco di posizioni di bit, la posizione del bit meno significativo è 0.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$bitsAllClear`operatore in Amazon DocumentDB.

Crea documenti di esempio

```
db.collection.insertMany([
  { _id: 1, bits: 0b1010 },
  { _id: 2, bits: 0b1100 },
  { _id: 3, bits: 0b0101 }
]);
```

Esempio di interrogazione

```
db.collection.find({
  bits: { $bitsAllClear: 0b0011 }
})
```

Output

```
{ "_id" : 2, "bits" : 12 }
```

La query verifica se tutti i bit specificati dalla maschera di bit `0b0011` (i due bit meno significativi) sono cancellati nel campo. `bits` Il documento con `_id` 2 soddisfa questa condizione, poiché tali bit `bits` sono stati cancellati nel campo.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$bitsAllClear` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$ bit AllSet

L'`$bitsAllSet`operatore in Amazon DocumentDB viene utilizzato per eseguire query su documenti in cui uno specifico set di bit è impostato su 1 in un campo. Questo operatore consente di eseguire

operazioni bit per bit sui valori dei campi e può essere utile quando è necessario controllare lo stato dei singoli bit all'interno di un valore numerico.

Parametri

- `field`: il nome del campo numerico su cui eseguire l'operazione bit per bit.
- `value`: La maschera di bit numerica che specifica quali bit devono essere controllati o un elenco di posizioni dei bit da controllare. Una maschera di bit numerica può essere in forma binaria (0b...), decimale, esadecimale (0x...), ottale (0o...) o binaria (). BinData In un elenco di posizioni di bit, la posizione del bit meno significativo è 0.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$bitsAllSet`operatore per trovare documenti in cui il `flags` campo ha tutti i bit impostati che sono specificati dalla maschera di bit.

Crea documenti di esempio

```
db.collection.insert([
  { _id: 1, flags: 0b1010 },
  { _id: 2, flags: 0b1100 },
  { _id: 3, flags: 0b1110 }
])
```

Esempio di interrogazione

```
db.collection.find({ flags: { $bitsAllSet: 0b1100 } })
```

Output

```
{ "_id": 2, "flags": 12 },
{ "_id": 3, "flags": 14 }
```

In questo esempio, la query verifica la presenza di documenti in cui nel `flags` campo sono impostati tutti i bit specificati dalla maschera di bit `0b1100` (che rappresenta il valore decimale 12). I documenti con `_id` 2 e 3 soddisfano questo criterio, poiché nei valori dei `flags` campi sono impostati tutti i bit richiesti (il terzo e il quarto bit meno significativi).

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$bitsAllSet` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$ bit AnyClear

L'`$bitsAnyClear`operatore in Amazon DocumentDB viene utilizzato per abbinare i documenti in cui uno qualsiasi dei bit specificati in un campo viene cancellato (impostato su 0). Questo può essere utile per eseguire operazioni bit per bit sui valori dei campi memorizzati nei documenti.

Parametri

- `field`: Il campo da controllare.
- `value`: La maschera di bit numerica che specifica quali bit devono essere controllati o un elenco di posizioni dei bit da controllare. Una maschera di bit numerica può essere in forma binaria (0b...), decimale, esadecimale (0x...), ottale (0o...) o binaria (). BinData In un elenco di posizioni di bit, la posizione del bit meno significativo è 0.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$bitsAnyClear`operatore per verificare se qualche bit è chiaro nel `status` campo della raccolta. `items`

Crea documenti di esempio

```
db.items.insertMany([
  { "_id": 1, "status": 7 },
  { "_id": 2, "status": 15 },
  { "_id": 3, "status": 31 }
]);
```

Esempio di interrogazione

```
db.items.find({ "status": { $bitsAnyClear: 8 } })
```

Output

```
{ "_id" : 1, "status" : 7 }
```

In questo esempio, la query verifica la presenza di documenti in cui il `status` campo contiene dei bit chiari (0) nella maschera di bit 8 (binaria`1000`). Il documento con `status` valori di 7 (binario`111`) corrisponde alla query, in quanto contiene almeno un bit clear nella maschera di bit fornita. Il clear bit corrispondente è il quarto bit meno importante.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$bitsAnyClear` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

$ bit AnySet

L'`$bitsAnySet`operatore in Amazon DocumentDB viene utilizzato per interrogare documenti in cui almeno un bit è impostato su 1 nei bit specificati in un campo. Questo operatore consente di eseguire operazioni bit a bit sui valori memorizzati nei campi, permettendo l'interrogazione e l'analisi efficienti dei dati con caratteristiche bit per bit.

Parametri

- `field`: nome del campo a cui applicare l'operazione bit per bit.
- `value`: La maschera di bit numerica che specifica quali bit devono essere controllati o un elenco di posizioni dei bit da controllare. Una maschera di bit numerica può essere in forma binaria (0b...), decimale, esadecimale (0x...), ottale (0o...) o binaria (). BinData In un elenco di posizioni di bit, la posizione del bit meno significativo è 0.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$bitsAnySet`operatore per trovare documenti in cui è impostato almeno un bit nel campo. `flags`

Crea documenti di esempio

```
db.collection.insertMany([
  { _id: 1, flags: 0b1010 },
  { _id: 2, flags: 0b1100 },
  { _id: 3, flags: 0b0011 },
  { _id: 4, flags: 0b0100 }
]);
```

Esempio di query

```
db.collection.find({
  flags: { $bitsAnySet: 0b1010 }
})
```

Output

```
{ "_id" : 1, "flags" : 10 }
{ "_id" : 2, "flags" : 12 }
```

```
{ "_id" : 3, "flags" : 3 }
```

La query restituisce i documenti in cui almeno uno dei bit specificati nella maschera di bit `0b1010` è impostato nel `flags` campo.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$bitsAnySet` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$comment`operatore in Amazon DocumentDB viene utilizzato per allegare un commento a una query. Questo può essere utile per fornire contesto o informazioni aggiuntive sulla query, utili per scopi di debug o documentazione. Il commento allegato verrà visualizzato come parte dell'output di operazioni come db.currentOp ().

Parametri

- `string`: Il commento allegato alla query.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$comment`operatore in Amazon DocumentDB.

Crea documenti di esempio

```
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" }
]);
```

Esempio di query

```
db.users.find({ age: { $gt: 25 } }, { _id: 0, name: 1, age: 1 }).comment("Retrieve
 users older than 25");
```

Output

```
{ "name" : "John Doe", "age" : 30 }
```

```
{ "name" : "Bob Johnson", "age" : 35 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$comment` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$elemMatch`operatore in Amazon DocumentDB viene utilizzato per interrogare un campo array e restituire documenti in cui almeno un elemento dell'array corrisponde ai criteri specificati. Questo operatore è particolarmente utile quando si dispone di strutture di dati complesse con array annidati o documenti incorporati.

La versione 2.0 di Planner ha aggiunto il supporto per gli indici per. `$elemMatch`

Parametri

- `field`: Il campo dell'array da interrogare.
- `query`: I criteri da confrontare con gli elementi dell'array.

Utilizzo **`$elemMatch`** all'interno di un'**`$all`**espressione

Vedi `Utilizzo di $elemMatch all'interno di un'espressione $all` le limitazioni relative all'uso dell'`$elemMatch`operatore all'interno di un'`$all`espressione.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$elemMatch`operatore per trovare documenti in cui l'`parts`array ha almeno un elemento che corrisponde ai criteri specificati.

Crea documenti di esempio

```
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 }] }
]);
```

Esempio di interrogazione

```
db.col.find({
  parts: { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } }
})
```

Output

```
{ "_id" : 1, "parts" : [ { "part" : "xyz", "qty" : 10 }, { "part" : "abc", "qty" :
 20 } ] }
{ "_id" : 2, "parts" : [ { "part" : "xyz", "qty" : 5 }, { "part" : "abc", "qty" :
 10 } ] }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$elemMatch` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$eq`operatore in Amazon DocumentDB viene utilizzato per abbinare documenti in cui il valore di un campo è uguale al valore specificato. Questo operatore è comunemente usato nel `find()` metodo per recuperare documenti che soddisfano i criteri specificati.

Parametri

- `field`: Il campo per verificare la condizione di uguaglianza.
- `value`: Il valore da confrontare con il campo.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$eq`operatore per trovare tutti i documenti in cui il `name` campo è uguale a. `"Thai Curry Palace"`

Crea documenti di esempio

```
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"] }
]);
```

Esempio di interrogazione

```
db.restaurants.find({ name: { $eq: "Thai Curry Palace" } });
```

Output

```
{ "_id" : ObjectId("68ee586f916df9d39f3d9414"), "name" : "Thai Curry Palace",
 "cuisine" : "Thai", "features" : [ "Private Dining" ] }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$eq` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$exists`operatore viene utilizzato per verificare se un campo esiste o meno in un documento. `$exists`particolarmente utile quando si lavora con indici sparsi in Amazon DocumentDB, dove il campo indicizzato potrebbe non essere presente in tutti i documenti.

Per utilizzare un indice Sparse creato in una query, devi utilizzare la clausola `$exists` nei campi che coprono l'indice. Se ometti`$exists`, Amazon DocumentDB non utilizzerà l'indice sparso per la query.

Parametri

- `field`: il nome del campo di cui verificare l'esistenza.
- `value`: Un valore booleano (`true`o`false`) che specifica se il campo deve esistere (`true`) o non esistere (`false`) nei documenti corrispondenti.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'uso dell'`$exists`operatore con un indice sparso sul `special_diets` campo della raccolta. `food`

Crea documenti di esempio

```
db.food.insertMany([
  { _id: 1, name: "Apple", special_diets: ["vegetarian", "gluten-free"] },
  { _id: 2, name: "Broccoli" },
  { _id: 3, name: "Chicken", special_diets: ["dairy-free"] }
]);
```

Crea un indice sparso nel campo `special_diets`

```
db.food.createIndex({ "special_diets": 1 }, { sparse: true, name:
 "special_diets_sparse_asc" });
```

Esempio di query

```
db.food.find({ "special_diets": { $exists: true } });
```

Output

```
[
  { "_id" : 1, "name" : "Apple", "special_diets" : [ "vegetarian", "gluten-free" ] },
  { "_id" : 3, "name" : "Chicken", "special_diets" : [ "dairy-free" ] }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$exists` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

Nuovo dalla versione 4.0.

Non supportato da Elastic Cluster.

L'`$expr`operatore in Amazon DocumentDB consente di utilizzare espressioni di aggregazione all'interno del linguaggio di interrogazione. Ti consente di eseguire confronti e calcoli complessi sui campi all'interno di un documento, in modo simile al modo in cui utilizzeresti le fasi della pipeline di aggregazione.

Gli operatori`$eq`, `$lt` `$lte``$gt`, e di `$gte` confronto inseriti in un `$expr` operatore possono utilizzare un indice, incluso un indice sulla `from` raccolta a cui si fa riferimento in una fase. `$lookup`

Limitazioni

- Gli indici possono essere utilizzati solo per i confronti tra campi e costanti, pertanto l'operando deve trasformarsi in una costante. Ad esempio, un confronto tra `$a` e un valore costante può utilizzare un indice, ma un confronto tra e non può. `$a` `$b`

- Gli indici a più chiavi, parziali o sparsi non vengono utilizzati. In questi casi, la query ritorna a una scansione della raccolta.
- Quando l'`let`operando restituisce un valore vuoto o mancante, il confronto non può utilizzare una ricerca mirata dell'indice. In questo caso, Amazon DocumentDB esegue una scansione completa dell'indice.

Parametri

- `expression`: un'espressione che restituisce un valore booleano, che consente di eseguire confronti e calcoli sui campi del documento.

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare l'`$expr`operatore per trovare tutti i documenti in cui il `manufacturingCost` campo è maggiore del campo. `price`

Crea documenti di esempio

```
db.inventory.insertMany([
  { item: "abc", manufacturingCost: 500, price: 100 },
  { item: "def", manufacturingCost: 300, price: 450 },
  { item: "ghi", manufacturingCost: 400, price: 120 }
]);
```

Esempio di interrogazione

```
db.inventory.find({
  $expr: {
    $gt: ["$manufacturingCost", "$price"]
  }
})
```

Output

```
{ "_id" : ObjectId("60b9d4d68d2cac581bc5a89a"), "item" : "abc", "manufacturingCost" :
 500, "price" : 100 },
{ "_id" : ObjectId("60b9d4d68d2cac581bc5a89c"), "item" : "ghi", "manufacturingCost" :
 400, "price" : 120 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$expr` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$gt`operatore in Amazon DocumentDB viene utilizzato per selezionare documenti in cui il valore del campo specificato è maggiore del valore specificato. Questo operatore è utile per filtrare e interrogare i dati sulla base di confronti numerici.

Parametri

- `field`: Il campo da confrontare.
- `value`: Il valore con cui effettuare il confronto.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$gt`operatore per trovare tutti i documenti in cui il `age` campo è maggiore di 30.

Crea documenti di esempio

```
db.users.insertMany([
  { name: "John", age: 25 },
  { name: "Jane", age: 32 },
  { name: "Bob", age: 45 },
  { name: "Alice", age: 28 }
]);
```

Esempio di query

```
db.users.find({ age: { $gt: 30 } });
```

Output

```
{ "_id" : ObjectId("6249e5c22a5d39884a0a0001"), "name" : "Jane", "age" : 32 },
```

```
{ "_id" : ObjectId("6249e5c22a5d39884a0a0002"), "name" : "Bob", "age" : 45 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$gt` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$gte`operatore in Amazon DocumentDB viene utilizzato per abbinare valori maggiori o uguali a un valore specificato. Questo operatore è utile per filtrare e interrogare i dati sulla base di confronti numerici.

Parametri

- `field`: Il campo da confrontare con il valore fornito.
- `value`: Il valore da confrontare con il campo.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$gte`operatore in Amazon DocumentDB per trovare tutti i documenti in cui il campo «età» è maggiore o uguale a 25.

Crea documenti di esempio

```
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 }
]);
```

Esempio di query

```
db.users.find({ age: { $gte: 25 } }, { _id: 0, name: 1, age: 1 });
```

Output

```
{ "name" : "Jane", "age" : 25 }
{ "name" : "Bob", "age" : 30 }
{ "name" : "Alice", "age" : 35 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$gte` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$in`operatore in Amazon DocumentDB è un operatore di interrogazione logica che consente di trovare documenti in cui il valore di un campo è uguale a uno qualsiasi dei valori specificati in un array.

Parametri

- `field`: il campo da confrontare con l'array fornito.
- `[value1, value2, ...]`: Una matrice di valori da confrontare con il campo specificato.

Dollaro (**$**) nei nomi dei campi

Dollaro ($) e punto (.) nei nomi dei campiPer le limitazioni relative all'interrogazione dei campi con $ prefisso `$in` negli oggetti annidati, vedere.

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare l'`$in`operatore per trovare documenti in cui il `color` campo è uno dei valori nell'array fornito.

Crea documenti di esempio

```
db.colors.insertMany([
  { "_id": 1, "color": "red" },
  { "_id": 2, "color": "green" },
  { "_id": 3, "color": "blue" },
  { "_id": 4, "color": "yellow" },
  { "_id": 5, "color": "purple" }
])
```

Esempio di interrogazione

```
db.colors.find({ "color": { "$in": ["red", "blue", "purple"] } })
```

Output

```
{ "_id": 1, "color": "red" },
{ "_id": 3, "color": "blue" },
```

```
{ "_id": 5, "color": "purple" }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$in` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

Novità dalla versione 4.0.

Non supportato dal cluster Elastic.

L'`$jsonSchema`operatore in Amazon DocumentDB viene utilizzato per filtrare i documenti in base a uno schema JSON specificato. Questo operatore consente di interrogare documenti che corrispondono a un particolare schema JSON, assicurando che i documenti recuperati rispettino requisiti strutturali e di tipo di dati specifici.

Utilizzando l'operatore di interrogazione di `$jsonSchema` valutazione come parte della creazione di una raccolta, è possibile convalidare lo schema dei documenti inseriti nella raccolta. Utilizzo della convalida dello schema JSONPer ulteriori informazioni, vedere.

Parametri

- `required`(array): specifica i campi obbligatori nel documento.
- `properties`(oggetto): Definisce il tipo di dati e altri vincoli per ogni campo del documento.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'uso dell'`$jsonSchema`operatore per filtrare la `employees` raccolta in modo da recuperare solo i documenti che contengono `age` i campi `employeeId` e e che il `name` `employeeId` campo è di tipo. `string`

Crea documenti di esempio

```
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" }
]);
```

Esempio di interrogazione

```
db.employees.aggregate([
  { $match: {
    $jsonSchema: {
      required: ["name", "employeeId", "age"],
      properties: { "employeeId": { "bsonType": "string" } }
    }
  }}
]);
```

Output

```
{ "_id" : ObjectId("6908e8b61f77fc26b2ecd26f"), "name" : { "firstName" : "Emily",
 "lastName" : "Brown" }, "employeeId" : "2", "age" : 25 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$jsonSchema` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$lt`operatore in Amazon DocumentDB viene utilizzato per abbinare valori inferiori al valore specificato. Questo operatore è utile per filtrare e interrogare i dati sulla base di confronti numerici.

Parametri

- `field`: nome del campo da controllare.
- `value`: Il valore con cui effettuare il confronto.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$lt`operatore per recuperare documenti il cui `Inventory.OnHand` valore è inferiore a 50.

Crea documenti di esempio

```
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"] }
]);
```

Esempio di query

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

Output

```
{ "_id" : 1, "Inventory" : { "OnHand" : 25, "Sold" : 60 }, "Colors" : [ "Red",
 "Blue" ] }
{ "_id" : 3, "Inventory" : { "OnHand" : 10, "Sold" : 85 }, "Colors" : [ "Red",
 "Green" ] }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$lt` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'operatore di $mod interrogazione seleziona i documenti in cui un valore di campo diviso per un divisore ha un resto specificato. Ciò è utile per filtrare i documenti in base alle condizioni aritmetiche del modulo.

Parametri

- `divisor`: Il numero per cui dividere.
- `remainder`: Il valore del resto previsto.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'$modoperatore per trovare tutti gli ordini in cui la quantità è un numero dispari.

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.orders.find({ quantity: { $mod: [2, 1] } });
```

Output

```
{ "_id" : 1, "item" : "Widget", "quantity" : 15 }
{ "_id" : 3, "item" : "Tool", "quantity" : 7 }
{ "_id" : 5, "item" : "Part", "quantity" : 9 }
```

Questa query restituisce documenti in cui la quantità divisa per 2 ha un resto di 1, selezionando efficacemente tutte le quantità dispari.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di $mod query, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$lte`operatore in Amazon DocumentDB viene utilizzato per abbinare documenti in cui il valore di un campo specificato è inferiore o uguale al valore specificato. Questo operatore è utile per filtrare e interrogare i dati sulla base di confronti numerici.

Parametri

- `field`: Il campo da confrontare.
- `value`: Il valore con cui effettuare il confronto.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$lte`operatore per recuperare documenti in cui il `quantity` campo è inferiore o uguale a 10.

Crea documenti di esempio

```
db.inventory.insertMany([
  { item: "canvas", qty: 100 },
  { item: "paint", qty: 50 },
  { item: "brush", qty: 10 },
  { item: "paper", qty: 5 }
]);
```

Esempio di interrogazione

```
db.inventory.find({ qty: { $lte: 10 } });
```

Output

```
{ "_id" : ObjectId("..."), "item" : "brush", "qty" : 10 },
{ "_id" : ObjectId("..."), "item" : "paper", "qty" : 5 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$lte` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$meta`operatore viene utilizzato per accedere ai metadati associati all'esecuzione corrente della query. Questo operatore viene utilizzato principalmente per le operazioni di ricerca di testo, in cui i metadati possono fornire informazioni sulla pertinenza dei documenti corrispondenti.

Parametri

- `textScore`: recupera il punteggio di ricerca testuale per il documento. Questo punteggio indica la pertinenza del documento rispetto alla query di ricerca testuale.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$meta`operatore per recuperare il punteggio di ricerca testuale per i documenti che corrispondono a una query di ricerca testuale.

Crea documenti di esempio

```
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." }
]);
```

Crea un indice di testo

```
db.documents.createIndex({ content: "text" });
```

Esempio di query

```
db.documents.find(
  { $text: { $search: "coffee" } },
  { _id: 0, title: 1, content: 1, score: { $meta: "textScore" } }
).sort({ score: { $meta: "textScore" } });
```

Output

```
[
  {
    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
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$meta` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$ne`operatore viene utilizzato per abbinare i documenti in cui il valore di un campo non è uguale al valore specificato. È un operatore di confronto che può essere utilizzato nei predicati di interrogazione per filtrare i documenti.

La versione 2.0 di Planner ha aggiunto il supporto all'indice per. `$ne`

Parametri

- `field`: Il campo da controllare.
- `value`: Il valore da confrontare.

Esempio (MongoDB Shell)

In questo esempio, troveremo tutti i documenti della `users` raccolta in cui il `status` campo non è uguale a. `"active"`

Crea documenti di esempio

```
db.users.insertMany([
  { name: "John", status: "active" },
  { name: "Jane", status: "inactive" },
  { name: "Bob", status: "suspended" },
  { name: "Alice", status: "active" }
]);
```

Esempio di interrogazione

```
db.users.find({ status: { $ne: "active" } });
```

Output

```
[
  {
    _id: ObjectId('...'),
    name: 'Jane',
    status: 'inactive'
  },
  {
```

```
    _id: ObjectId('...'),
    name: 'Bob',
    status: 'suspended'
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del $ne comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$nin`operatore viene utilizzato per abbinare valori che non sono presenti nell'array specificato. È l'inverso dell'`$in`operatore, che corrisponde ai valori presenti nell'array specificato.

La versione 2.0 di Planner ha aggiunto il supporto per gli indici per. `$nin`

Parametri

- `field`: Il campo da controllare.
- `array`: La matrice di valori da confrontare.

Dollar (**$**) nei nomi dei campi

Vedi le limitazioni Dollaro ($) e punto (.) nei nomi dei campi relative all'interrogazione di campi con $ prefisso `$nin` negli oggetti annidati.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$nin`operatore per trovare documenti in cui il `category` campo non è uguale a «Fiction» o «Mystery».

Crea documenti di esempio

```
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" }
])
```

Esempio di interrogazione

```
db.books.find({
  category: {
```

```
    $nin: ["Fiction", "Mystery"]
  }
})
```

Output

```
[
  {
    _id: ObjectId('...'),
    title: 'The Martian',
    author: 'Andy Weir',
    category: 'Science Fiction'
  },
  {
    _id: ObjectId('...'),
    title: 'The Alchemist',
    author: 'Paulo Coelho',
    category: 'Philosophy'
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$nin` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$nor`operatore viene utilizzato per abbinare documenti in cui nessuna delle condizioni di interrogazione specificate è vera. È simile all'operazione logica «NOR», in cui il risultato è vero se nessuno degli operandi è vero.

Parametri

- `expression1`: La prima espressione da valutare.
- `expression2`: La seconda espressione da valutare.
- `expressionN`: espressioni aggiuntive da valutare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$nor`operatore recuperando documenti in cui il `qty` campo non è inferiore a 20 e il `size` campo non è uguale a «XL».

Crea documenti di esempio

```
db.items.insertMany([
  { qty: 10, size: "M" },
  { qty: 15, size: "XL" },
  { qty: 25, size: "L" },
  { qty: 30, size: "XL" }
])
```

Esempio di interrogazione

```
db.items.find({
  $nor: [
    { qty: { $lt: 20 } },
    { size: "XL" }
  ]
})
```

Output

```
[
  { "_id" : ObjectId("..."), "qty" : 25, "size" : "L" }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$nor` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$not`operatore viene utilizzato per negare il risultato di una determinata espressione. Consente di selezionare documenti in cui la condizione specificata non corrisponde.

La versione 2.0 di Planner ha aggiunto il supporto degli indici per `$not {eq}` e`$not {in}`.

Parametri

- `expression`: L'espressione da negare.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$not`operatore per trovare documenti in cui il `status` campo non è uguale a «attivo».

Crea documenti di esempio

```
db.users.insertMany([
  { name: "John", status: "active" },
  { name: "Jane", status: "inactive" },
  { name: "Bob", status: "pending" },
  { name: "Alice", status: "active" }
]);
```

Esempio di query

```
db.users.find({ status: { $not: { $eq: "active" } } });
```

Output

```
[
  {
    _id: ObjectId('...'),
    name: 'Jane',
    status: 'inactive'
  },
  {
    _id: ObjectId('...'),
    name: 'Bob',
    status: 'pending'
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$not` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$or`operatore viene utilizzato per eseguire un'operazione OR logica su una matrice di due o più espressioni. Restituisce documenti che corrispondono ad almeno una delle espressioni. Questo operatore è utile quando è necessario ricercare documenti che soddisfano una delle molteplici condizioni.

Parametri

- `expression1`: la prima espressione da valutare.
- `expression2`: La seconda espressione da valutare.
- `...`: espressioni aggiuntive da valutare (opzionale).

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$or`operatore per trovare documenti in cui si `make` trova "" con il modello «Heavy TruckForYou H1" o" "con il modello «Bolid SportForYou 1".

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.cars.find({
  $or: [
    { make: "TruckForYou", model: "Heavy H1" },
    { make: "SportForYou", model: "Bolid 1" }
  ]
});
```

Output

```
[
```

```
  {
    _id: ObjectId('...'),
    make: 'TruckForYou',
    model: 'Heavy H1',
    year: 2020
  },
  {
    _id: ObjectId('...'),
    make: 'SportForYou',
    model: 'Bolid 1',
    year: 2021
  }
]
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$or` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$regex`operatore consente di eseguire la corrispondenza delle espressioni regolari nei campi stringa. È un potente strumento per la ricerca e il filtraggio di documenti in base a modelli complessi.

Parametri

- `regular expression`: Il modello di espressione regolare da confrontare con il campo.
- `$options`: (opzionale) Fornisce opzioni per modificare il comportamento di ricerca, ad esempio la distinzione tra maiuscole e minuscole, la corrispondenza globale, ecc.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$regex`operatore per cercare documenti in cui il campo «nome» corrisponde a uno schema specifico.

Crea documenti di esempio

```
db.users.insertMany([
  { name: "John Doe" },
  { name: "Jane Smith" },
  { name: "Alice Johnson" },
  { name: "Bob Williams" },
  { name: "Charlie Davis" }
]);
```

Esempio di query

```
db.users.find({ name: { $regex: /^A/ } })
```

Output

```
[
  { "_id" : ObjectId("..."), "name" : "Alice Johnson" }
]
```

Questa query restituirà tutti i documenti in cui il campo «nome» inizia con la lettera «A».

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$regex` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

$Frequenza di campionamento

Nuovo dalla versione 8.0.1.

Non supportato da Elastic Cluster.

L'`$sampleRate`operatore in Amazon DocumentDB corrisponde a un campione casuale dei documenti di input in base alla frequenza specificata. Puoi utilizzarlo in un filtro di `find()` interrogazione o in una fase di aggregazione. `$match` Poiché la selezione è probabilistica, il numero di documenti restituiti è approssimativo e può variare tra le esecuzioni.

Parametri

- `value`: un numero compreso nell'intervallo `[0, 1]` (incluso), passato direttamente all'operatore (ad esempio`{ $sampleRate: 0.3 }`), che imposta la probabilità di includere ogni documento. Un valore di non `0` restituisce alcun documento; un valore di `1` restituisce tutti i documenti.

Esempio (MongoDB Shell)

L'esempio seguente viene utilizzato `$sampleRate` per restituire circa il 30 percento dei documenti in una raccolta.

Crea documenti di esempio

```
db.events.insertMany([
  { _id: 1, type: "click" },
  { _id: 2, type: "view" },
  { _id: 3, type: "click" },
  { _id: 4, type: "view" },
  { _id: 5, type: "purchase" }
]);
```

Esempio di interrogazione

```
db.events.aggregate([
  { $match: { $sampleRate: 0.3 } }
]);
```

Output

L'operazione restituisce un sottoinsieme casuale dei documenti. Poiché `$sampleRate` è probabilistico, i documenti specifici restituiti e il loro numero variano ogni volta che viene eseguita la query.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$sampleRate` comando, scegliete la scheda corrispondente alla lingua che desiderate utilizzare:

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

L'operatore di `$slice` proiezione limita il numero di elementi dell'array restituiti nei risultati di una query. Consente di recuperare un numero specifico di elementi dall'inizio o dalla fine di un campo dell'array senza caricare l'intero array.

Parametri

- `field`: il campo dell'array da proiettare.
- `count`: numero di elementi da restituire. I valori positivi restituiscono gli elementi dall'inizio, i valori negativi dalla fine.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'operatore di `$slice` proiezione per restituire solo i primi due elementi da un campo di matrice.

Create documenti di esempio

```
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"] }
]);
```

Esempio di interrogazione

```
db.inventory.find(
  {},
  { item: 1, tags: { $slice: 2 } }
)
```

Output

```
{ "_id" : 1, "item" : "notebook", "tags" : [ "office", "school" ] }
{ "_id" : 2, "item" : "pen", "tags" : [ "office", "writing" ] }
{ "_id" : 3, "item" : "folder", "tags" : [ "office", "supplies" ] }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di `$slice` proiezione, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'operatore di `$size` interrogazione corrisponde ai documenti in cui un campo array contiene esattamente il numero di elementi specificato. Questo è utile per filtrare i documenti in base alla lunghezza dell'array.

Parametri

- `field`: Il campo dell'array da controllare.
- `count`: Il numero esatto di elementi che l'array deve contenere.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$size`operatore per trovare tutti i prodotti che hanno esattamente tre tag.

Crea documenti di esempio

```
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"] }
]);
```

Esempio di interrogazione

```
db.products.find({ tags: { $size: 3 } });
```

Output

```
{ "_id" : 1, "name" : "Laptop", "tags" : [ "electronics", "computers", "portable" ] }
{ "_id" : 3, "name" : "Desk", "tags" : [ "furniture", "office", "workspace" ] }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore di `$size` query, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$text`operatore viene utilizzato per eseguire una ricerca full-text su campi indicizzati di testo all'interno di una raccolta di documenti. Questo operatore consente di cercare documenti che contengono parole o frasi specifiche e può essere combinato con altri operatori di interrogazione per filtrare i risultati in base a criteri aggiuntivi.

Parametri

- `$search`: la stringa di testo da cercare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare l'`$text`operatore per cercare documenti contenenti la parola «interesse» e filtrare i risultati in base a un campo «star_rating».

Crea documenti di esempio

```
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" }
]);
```

Crea un indice di testo

```
db.test.createIndex({ comments: "text" });
```

Esempio di interrogazione

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "interest"}}]})
```

Output

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
{ "_id" : 6, "star_rating" : 5, "comments" : "interested in couch for sale, year
 2022" }
```

Il comando precedente restituisce documenti con un campo indicizzato di testo contenente qualsiasi forma di «interesse» e un «star_rating» uguale a 5.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$text` comando, scegli la scheda corrispondente alla lingua che desideri utilizzare:

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

L'`$type`operatore viene utilizzato per controllare il tipo di dati di un campo in un documento.
Può essere utilizzato quando sono necessarie operazioni o convalide specifiche per tipo.
L'`$type`operatore restituisce il tipo BSON dell'espressione valutata. Il tipo restituito è una stringa, che corrisponde al tipo di campo o espressione.

La versione 2.0 di Planner ha aggiunto il supporto per gli indici per`$type`.

Parametri

- `expression`: L'espressione da valutare.

Esempio (MongoDB Shell)

Crea documenti di esempio

```
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 }
]);
```

Esempio di interrogazione

```
db.documents.find({
  $or: [
    { age: { $type: "number" } },
    { email: { $type: "string" } },
```

```
    { name: { $type: "string" } }
  ]
})
```

Output

```
[
  { "_id": 1, "name": "John", "age": 30, "email": "john@example.com" },
  { "_id": 2, "name": "Jane", "age": "25", "email": 123456 }
]
```

Questa query restituirà i documenti in cui il `age` campo è di tipo «numero», il `email` campo è di tipo «stringa» e il `name` campo è di tipo «stringa».

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$type` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

Aggiorna operatori

Questa sezione fornisce informazioni dettagliate sugli operatori di aggiornamento supportati da Amazon DocumentDB.

Argomenti

- $
- $[]
- $ [<identificatore>]
- $aggiungi ToSet
- $bit
- $currentDate
- $each

- $inc
- $max
- $min
- $mul
- $pop
- $position
- $pull
- $pullAll
- $push
- $rename
- $set
- $set OnInsert
- $slice
- $sort
- $unset

$

L'operatore $ posizionale aggiorna il primo elemento dell'array che corrisponde alla condizione di interrogazione. Funge da segnaposto per la posizione dell'elemento dell'array corrispondente.

Parametri

- `field.$`: Il campo dell'array con l'operatore posizionale per aggiornare il primo elemento corrispondente.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'operatore $ posizionale per aggiornare un elemento specifico dell'array.

Crea documenti di esempio

```
db.inventory.insertMany([
  { _id: 1, item: "Widget", quantities: [10, 20, 30] },
  { _id: 2, item: "Gadget", quantities: [5, 15, 25] }
```

```
]);
```

Esempio di interrogazione

```
db.inventory.updateOne(
  { _id: 1, quantities: 20 },
  { $set: { "quantities.$": 22 } }
);
```

Output

```
{
  "_id" : 1,
  "item" : "Widget",
  "quantities" : [ 10, 22, 30 ]
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore $ posizionale, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$[]

L'operatore `$[]` all posizionale aggiorna tutti gli elementi di un array. Viene utilizzato quando è necessario modificare ogni elemento in un campo dell'array.

Parametri

- `field.$[]`: Il campo array con l'operatore all-posizionale per aggiornare tutti gli elementi.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$[]`operatore per aggiornare tutti gli elementi dell'array.

Crea documenti di esempio

```
db.products.insertOne({
  _id: 1,
  name: "Laptop",
  prices: [1000, 1100, 1200]
});
```

Esempio di query

```
db.products.updateOne(
  { _id: 1 },
  { $inc: { "prices.$[]": 50 } }
);
```

Output

```
{
  "_id" : 1,
  "name" : "Laptop",
  "prices" : [ 1050, 1150, 1250 ]
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$[]` dell'operatore, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

$ [<identificatore>]

L'operatore posizionale `$[<identifier>]` filtrato aggiorna tutti gli elementi dell'array che soddisfano le condizioni di filtro specificate. Viene utilizzato con l'`arrayFilters`opzione di aggiornare selettivamente gli elementi dell'array.

Parametri

- `field.$[identifier]`: Il campo dell'array con l'operatore posizionale filtrato.
- `arrayFilters`: una serie di condizioni di filtro che determinano gli elementi da aggiornare.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$[<identifier>]`operatore per aggiornare elementi specifici dell'array in base a una condizione.

Crea documenti di esempio

```
db.students.insertOne({
  _id: 1,
  name: "Alice",
  grades: [
    { subject: "Math", score: 85 },
    { subject: "Science", score: 92 },
    { subject: "History", score: 78 }
  ]
});
```

Esempio di interrogazione

```
db.students.updateOne(
  { _id: 1 },
  { $inc: { "grades.$[elem].score": 5 } },
  { arrayFilters: [{ "elem.score": { $gte: 80 } }] }
);
```

Output

```
{
  "_id" : 1,
  "name" : "Alice",
  "grades" : [
    { "subject" : "Math", "score" : 90 },
    { "subject" : "Science", "score" : 97 },
    { "subject" : "History", "score" : 78 }
  ]
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo `$[<identifier>]` dell'operatore, scegli la scheda relativa alla lingua che desideri utilizzare:

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()
```

$aggiungi ToSet

L'`$addToSet`operatore in Amazon DocumentDB viene utilizzato per aggiungere un valore a un array solo se il valore non è già presente nell'array. Ciò è utile per garantire che un array contenga elementi unici.

Parametri

- `field`: Il campo da aggiornare.
- `value`: Il valore da aggiungere al campo dell'array. Può essere un singolo valore o un'espressione.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$addToSet`operatore per aggiungere elementi unici a un array.

Crea documenti di esempio

```
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"] }
])
```

Esempio di interrogazione

```
db.products.update(
  { "item": "apple" },
  { $addToSet: { "tags": "green" } }
)
```

Output

```
{ "_id": 1, "item": "apple", "tags": ["fruit", "red", "round", "green"] }
```

In questo esempio, l'`$addToSet`operatore aggiunge il tag «verde» all'array «tags» del documento in cui il campo «item» è «apple». Poiché «green» non era già presente nell'array, è stato aggiunto.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$addToSet` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$bit`operatore in Amazon DocumentDB consente di eseguire operazioni bit per bit sui bit di un determinato campo. Questo può essere utile per attività come l'impostazione, la cancellazione o il controllo dello stato dei singoli bit all'interno di un numero.

Parametri

- `field`: Il campo su cui eseguire operazioni bit per bit.
- `and`: Un valore intero utilizzato per eseguire un'operazione AND bit per bit sul campo.
- `or`: un valore intero utilizzato per eseguire un'operazione OR bit per bit sul campo.
- `xor`: Un valore intero utilizzato per eseguire un'operazione XOR bit per bit sul campo.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$bit`operatore per eseguire operazioni bit per bit su un campo numerico.

Crea documenti di esempio

```
db.numbers.insert([
  { "_id": 1, "number": 5 },
  { "_id": 2, "number": 12 }
])
```

Esempio di interrogazione

```
db.numbers.update(
  { "_id": 1 },
  { "$bit": { "number": { "and": 3 } } }
)
```

Output

```
{
  "_id": 1,
  "number": 1
}
```

In questo esempio, l'`$bit`operatore viene utilizzato per eseguire un'operazione AND bit per bit sul campo «numero» del documento con il valore `_id` di 1. Il risultato è che il valore del campo «numero» è impostato su 1, che è il risultato dell'operazione AND bit per bit tra il valore originale di 5 e il valore 3.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$bit` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$currentDate`operatore viene utilizzato per impostare il valore di un campo sulla data e l'ora correnti. Questo operatore è utile per aggiornare automaticamente un campo con il timestamp corrente quando un documento viene inserito o aggiornato.

Parametri

- `field`: Il campo da aggiornare con la data e l'ora correnti.
- `type`: (opzionale) Specificate il tipo BSON da utilizzare per la data corrente. Può essere `date` o `timestamp`.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$currentDate`operatore per impostare il `lastModified` campo sulla data e l'ora correnti quando viene inserito un nuovo documento.

Crea documenti di esempio

```
db.users.insert({
  name: "John Doe",
  email: "john.doe@example.com"
})
```

Esempio di query

```
db.users.updateOne(
  { name: "John Doe" },
  { $currentDate: { lastModified: true } }
)
```

Visualizza il documento aggiornato

```
db.users.findOne({ name: "John Doe" })
```

Output

```
{
  _id: ObjectId('...'),
  name: 'John Doe',
  email: 'john.doe@example.com',
  lastModified: ISODate('2025-10-25T22:50:29.963Z')
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$currentDate` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$each`operatore viene utilizzato insieme ad altri operatori di aggiornamento, come `$push` e`$addToSet`, per aggiungere più valori a un campo di matrice. Consente di aggiungere più elementi a un array in un'unica operazione, anziché dover eseguire più operazioni di aggiornamento.

Parametri

- `value`: La matrice di valori da aggiungere al campo dell'array.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo `$each` dell'operatore con l'`$push`operatore per aggiungere più elementi a un campo di matrice.

Crea documenti di esempio

```
db.fruits.insertOne({
  _id: 1,
  fruits: ["apple", "banana"]
})
```

Esempio di query

```
db.fruits.updateOne(
  { _id: 1 },
  { $push: { fruits: { $each: ["cherry", "durian", "elderberry"] } } }
)
```

Visualizza il documento aggiornato

```
db.fruits.findOne({ _id: 1 })
```

Output

```
{
  _id: 1,
  fruits: [ 'apple', 'banana', 'cherry', 'durian', 'elderberry' ]
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$each` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$inc`operatore viene utilizzato per incrementare il valore di un campo di un importo specificato. Viene utilizzato per aggiornare un campo numerico, ad esempio un contatore o una valutazione, senza dover recuperare il valore corrente, calcolare il nuovo valore e quindi aggiornare il campo.

Parametri

- `field`: nome del campo da incrementare.
- `amount`: La quantità di cui incrementare il campo. Può essere un valore positivo o negativo.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$inc`operatore per incrementare il `age` campo di un documento.

Crea documenti di esempio

```
db.users.insertOne({_id: 123, name: "John Doe", age: 30})
```

Esempio di interrogazione

```
db.users.updateOne({_id: 123}, {$inc: {age: 1}})
```

Visualizza il documento aggiornato

```
db.users.findOne({_id: 123})
```

Output

```
{ "_id" : 123, "name" : "John Doe", "age" : 31 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$inc` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'operatore `$max` update aggiorna il valore di un campo solo se il valore specificato è maggiore del valore del campo corrente. Questo operatore è utile per mantenere i valori massimi tra gli aggiornamenti.

Parametri

- `field`: Il campo da aggiornare.
- `value`: Il valore da confrontare con il valore del campo corrente.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo `$max` dell'operatore per aggiornare il punteggio più alto registrato per un giocatore.

Crea documenti di esempio

```
db.scores.insertMany([
  { _id: 1, player: "Alice", highScore: 85 },
  { _id: 2, player: "Bob", highScore: 92 },
  { _id: 3, player: "Charlie", highScore: 78 }
])
```

Esempio di aggiornamento

```
db.scores.updateOne(
  { _id: 1 },
  { $max: { highScore: 95 } }
)
```

Risultato

Il `highScore` campo per Alice viene aggiornato a 95 perché 95 è maggiore del valore corrente di 85.

```
{ "_id": 1, "player": "Alice", "highScore": 95 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$max` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'operatore `$min` update aggiorna il valore di un campo solo se il valore specificato è inferiore al valore del campo corrente. Questo operatore è utile per mantenere i valori minimi tra gli aggiornamenti.

Parametri

- `field`: Il campo da aggiornare.
- `value`: Il valore da confrontare con il valore del campo corrente.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo `$min` dell'operatore per aggiornare la temperatura più bassa registrata per una stazione meteorologica.

Crea documenti di esempio

```
db.weather.insertMany([
  { _id: 1, station: "Station A", lowestTemp: 15 },
  { _id: 2, station: "Station B", lowestTemp: 20 },
  { _id: 3, station: "Station C", lowestTemp: 18 }
])
```

Esempio di aggiornamento

```
db.weather.updateOne(
  { _id: 1 },
  { $min: { lowestTemp: 12 } }
)
```

Risultato

Il `lowestTemp` campo per la stazione A viene aggiornato a 12 perché 12 è inferiore al valore corrente di 15.

```
{ "_id": 1, "station": "Station A", "lowestTemp": 12 }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$min` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$mul`operatore in Amazon DocumentDB viene utilizzato per moltiplicare il valore di un campo per un numero specificato. Ciò può essere utile per aggiornare più documenti in modo atomico e coerente, ad esempio per aggiornare le miglia di volo in base allo stato di una carta di credito.

Parametri

- `field`: Il campo da moltiplicare.
- `multiplier`: Il numero per cui moltiplicare il valore del campo.

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'`$mul`operatore per raddoppiare il `flight_miles` valore di tutti i documenti in cui si trova il `credit_card` campo. `true`

Crea documenti di esempio

```
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] }
]);
```

Esempio di query

```
db.miles.update(
  { "credit_card": { "$eq": true } },
```

```
  { "$mul": { "flight_miles.$[]": NumberInt(2) } },
  { "multi": true }
);
```

Output

```
{ "_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 ] }
```

Per i clienti che dispongono di una carta di credito, le miglia di volo sono state raddoppiate.

L'operatore di matrice `$[]` posizionale viene utilizzato per applicare l'`$mul`operazione a ciascun elemento dell'array. `flight_miles`

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$mul` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'$popoperatore in Amazon DocumentDB viene utilizzato per rimuovere il primo o l'ultimo elemento da un campo array. È particolarmente utile quando è necessario mantenere un array a dimensione fissa o implementare una struttura di dati simile a una coda all'interno di un documento.

Parametri

- `field`: il nome del campo dell'array da cui rimuovere un elemento.
- `value`: un valore intero che determina la posizione dell'elemento da rimuovere. Un valore di `1` rimuove l'ultimo elemento, mentre un valore di `-1` rimuove il primo elemento.

Esempio (MongoDB Shell)

Questo esempio dimostra come utilizzare l'$popoperatore per rimuovere il primo e l'ultimo elemento da un campo di matrice.

Crea documenti di esempio

```
db.users.insertMany([
  { "_id": 1, "name": "John Doe", "hobbies": ["reading", "swimming", "hiking"] },
  { "_id": 2, "name": "Jane Smith", "hobbies": ["cooking", "gardening", "painting"] }
])
```

Esempio di interrogazione

```
// 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 } })
```

Output

```
{ "_id" : 1, "name" : "John Doe", "hobbies" : [ "swimming", "hiking" ] }
{ "_id" : 2, "name" : "Jane Smith", "hobbies" : [ "cooking", "gardening" ] }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del $pop comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

Il `$position` modificatore in Amazon DocumentDB specifica la posizione nell'array in cui l'operatore inserisce `$push` gli elementi. Senza il `$position` modificatore, l'`$push`operatore inserisce gli elementi alla fine dell'array.

Parametri

- `field`: il campo dell'array da aggiornare.
- `num`: la posizione nell'array in cui gli elementi devono essere inseriti, in base all'indicizzazione a base zero.

Nota: per utilizzare il `$position` modificatore, deve apparire insieme al modificatore. `$each`

Esempio (MongoDB Shell)

L'esempio seguente dimostra come utilizzare l'`$position`operatore per inserire attività in posizioni specifiche in un sistema di gestione dei progetti.

Crea documenti di esempio

```
db.projects.insertOne({ "_id": 1, "name": "Website Redesign", "tasks": ["Design
 mockups"] })
```

Esempio di query 1: aggiungi attività urgenti all'inizio

```
db.projects.updateOne(
   { _id: 1 },
   {
     $push: {
        tasks: {
           $each: ["Security audit", "Performance review"],
           $position: 0
        }
     }
   }
)
```

Uscita 1

```
{ "_id": 1, "name": "Website Redesign", "tasks": ["Security audit", "Performance
 review", "Design mockups"] }
```

Esempio di query 2: aggiungere attività in una posizione specifica

```
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
        }
     }
   }
)
```

Output 2

```
{ "_id": 2, "name": "Mobile App", "tasks": ["Setup project", "Create wireframes", "Code
 review", "Testing phase", "Deploy to store"] }
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$position` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$pull`operatore viene utilizzato per rimuovere da un array tutte le istanze di uno o più valori che corrispondono a una condizione specificata. Questo operatore è utile quando è necessario rimuovere elementi specifici da un campo di matrice all'interno di un documento.

Parametri

- `field`: il nome del campo dell'array da cui rimuovere i valori.
- `value`: il valore o la condizione che determina quali elementi rimuovere dall'array.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$pull`operatore per rimuovere elementi da un campo di matrice.

Crea documenti di esempio

```
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"]
  }
])
```

Esempio di interrogazione

```
db.restaurants.updateMany(
  { cuisine: "Italian" },
```

```
  { $pull: { features: "Takeout" } }
)
```

Output

```
{
  "acknowledged" : true,
  "matchedCount" : 1,
  "modifiedCount" : 1
}
```

La query precedente rimuove la funzione «Takeout» da tutti i documenti in cui il `cuisine` campo è «italiano».

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$pull` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$pullAll`operatore in Amazon DocumentDB viene utilizzato per rimuovere tutte le istanze dei valori specificati da un campo array. Ciò è particolarmente utile quando è necessario rimuovere più elementi da un array in un'unica operazione.

Parametri

- `field`: il nome del campo dell'array da cui rimuovere gli elementi.
- `value`: Una matrice di valori da rimuovere dal campo dell'array.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$pullAll`operatore per rimuovere più elementi da un campo di matrice.

Crea documenti di esempio

```
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"]
  }
])
```

Esempio di interrogazione

```
db.restaurants.update(
  { "name": "Taj Mahal" },
  { $pullAll: { "features": ["Private Dining", "Live Music"] } }
)
```

Output

```
{
  "name": "Taj Mahal",
  "cuisine": "Indian",
  "features": []
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$pullAll` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$push`operatore in Amazon DocumentDB viene utilizzato per aggiungere un elemento a un campo array in un documento. Questo operatore è particolarmente utile quando è necessario aggiungere nuovi dati a un array esistente senza sovrascrivere l'intero array.

Parametri

- `field`: il nome del campo dell'array a cui aggiungere il nuovo elemento.
- `value`: Il valore da aggiungere all'array.
- `position`: (opzionale) Un modificatore che specifica la posizione nell'array in cui deve essere aggiunto il nuovo elemento. I modificatori supportati includono $ (aggiungi alla fine dell'array) e `$[]` (aggiungi alla fine dell'array, ignorando qualsiasi filtro dell'array).

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$push`operatore per aggiungere nuovi elementi a un campo di matrice in un documento.

Crea documenti di esempio

```
db.users.insert([
  { _id: 1, name: "John Doe", hobbies: ["reading", "swimming"] },
  { _id: 2, name: "Jane Smith", hobbies: ["gardening", "cooking"] }
])
```

Esempio di interrogazione

```
db.users.updateOne(
  { _id: 1 },
  { $push: { hobbies: "hiking" } }
)
```

Output

```
{
  "acknowledged" : true,
  "matchedCount" : 1,
  "modifiedCount" : 1
```

```
}
```

Dopo aver eseguito l'aggiornamento, il documento con `_id: 1` avrà l'`hobbies`array aggiornato `a["reading", "swimming", "hiking"]`.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$push` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$rename`operatore in Amazon DocumentDB viene utilizzato per rinominare un campo in un documento. Questo operatore può essere particolarmente utile quando è necessario aggiornare la struttura dei documenti o allinearli a nuovi modelli di dati.

Parametri

- `field`: Il campo da rinominare.
- `newName`: Il nuovo nome per il campo.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$rename`operatore per rinominare il `"Date.DoW"` campo `"Date.DayOfWeek"` in un documento con il `"DocName"` campo impostato su. `"Document 1"`

Crea documenti di esempio

```
db.example.insertOne({
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    },
    "Words": 2482
})
```

Esempio di interrogazione

```
db.example.update(
    { "DocName": "Document 1" },
    { $rename: { "Date.DoW": "Date.DayOfWeek" } }
)
```

Output

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

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$rename` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

L'`$set`operatore in Amazon DocumentDB viene utilizzato per aggiornare il valore di un campo specificato in un documento. Questo operatore consente di aggiungere nuovi campi o modificare quelli esistenti all'interno di un documento. È un operatore di aggiornamento fondamentale nel driver Java MongoDB, compatibile con Amazon DocumentDB.

Parametri

- `field`: Il campo da aggiornare.
- `value`: Il nuovo valore per il campo.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$set`operatore per aggiornare il `Item` campo in un documento.

Crea documenti di esempio

```
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
    }
  }
])
```

Esempio di query

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

Output

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

```
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$set` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

$set OnInsert

L'`$setOnInsert`operatore in Amazon DocumentDB viene utilizzato per impostare il valore di un campo se viene inserito un documento, ma non ha effetto se il documento viene aggiornato.

Parametri

- `field`: Il campo da impostare.
- `value`: Il valore da assegnare al campo.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo dell'`$setOnInsert`operatore in Amazon DocumentDB. Crea un nuovo documento se il documento non esiste già, ma non ha effetto se il documento viene aggiornato.

Crea documenti di esempio

```
db.users.insertOne({
  _id: 1,
  name: "John Doe",
  age: 30
})
```

Esempio di query 1: aggiorna un documento esistente

```
db.users.update(
  { _id: 1 },
  {
    $set: { age: 31 },
```

```
    $setOnInsert: { createdAt: new Date() }
  },
  { upsert: true }
)
```

Output 1

```
{
  _id: 1,
  name: "John Doe",
  age: 31
}
```

L'output mostra che il documento è stato aggiornato, ma il `createdAt` campo NON è stato aggiunto poiché il documento esisteva già. L'`$setOnInsert`operatore si applica solo quando si inseriscono nuovi documenti.

Esempio di query 2 - Inserisci nuovo documento (upsert)

```
db.users.update(
  { _id: 2 },
  {
    $set: { name: "Jane Smith", age: 25 },
    $setOnInsert: { createdAt: new Date() }
  },
  { upsert: true }
)
```

Uscita 2

```
{
  _id: 2,
  name: "Jane Smith",
  age: 25,
  createdAt: ISODate("2025-10-31T09:57:52.459Z")
}
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$setOnInsert` comando, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'operatore di `$slice` aggiornamento modifica un array limitandone le dimensioni. Se utilizzato con l'`$push`operatore, limita il numero di elementi in un array, mantenendo solo il numero specificato di elementi più recenti o meno recenti.

Parametri

- `field`: il campo dell'array da modificare.
- `count`: numero massimo di elementi da conservare. I valori positivi mantengono i primi N elementi, i valori negativi mantengono gli ultimi N elementi.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'operatore `$slice` update with per mantenere un array `$push` a dimensione fissa di punteggi recenti.

Crea documenti di esempio

```
db.students.insertOne({
  _id: 1,
  name: "Alice",
  scores: [85, 90, 78]
});
```

Esempio di interrogazione

```
db.students.updateOne(
```

```
  { _id: 1 },
  {
    $push: {
      scores: {
        $each: [92, 88],
        $slice: -3
      }
    }
  }
)
```

Output

```
{
  "_id" : 1,
  "name" : "Alice",
  "scores" : [ 78, 92, 88 ]
}
```

In questo esempio, il `$slice: -3` modificatore mantiene solo gli ultimi tre elementi dopo aver inserito nuovi valori nell'array.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo dell'operatore `$slice` update, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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

Il modificatore di `$sort` aggiornamento ordina gli elementi dell'array quando viene utilizzato con l'`$push`operatore. Dispone gli elementi dell'array in ordine crescente o decrescente in base ai valori di campo specificati o agli elementi stessi.

Parametri

- `field`: il campo dell'array da modificare.
- `order`: da utilizzare 1 per l'ordine crescente o `-1` per l'ordine decrescente.

Esempio (MongoDB Shell)

L'esempio seguente dimostra l'utilizzo del `$sort` modificatore con `$push` per aggiungere nuovi punteggi dei quiz e mantenerli ordinati in ordine decrescente.

Crea documenti di esempio

```
db.students.insertOne({
  _id: 1,
  name: "Bob",
  quizzes: [
    { score: 85, date: "2024-01-15" },
    { score: 92, date: "2024-02-10" }
  ]
});
```

Esempio di query

```
db.students.updateOne(
  { _id: 1 },
  {
    $push: {
      quizzes: {
        $each: [{ score: 78, date: "2024-03-05" }],
        $sort: { score: -1 }
      }
    }
```

```
  }
)
```

Output

```
{
  "_id" : 1,
  "name" : "Bob",
  "quizzes" : [
    { "score" : 92, "date" : "2024-02-10" },
    { "score" : 85, "date" : "2024-01-15" },
    { "score" : 78, "date" : "2024-03-05" }
  ]
}
```

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del modificatore di `$sort` aggiornamento, scegli la scheda relativa alla lingua che desideri utilizzare:

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

L'`$unset`operatore in Amazon DocumentDB viene utilizzato per rimuovere un campo specificato da un documento. Quando un campo viene rimosso utilizzando`$unset`, il campo viene eliminato

dal documento e le dimensioni del documento vengono ridotte di conseguenza. Ciò può essere utile quando si desidera rimuovere dati non necessari dai documenti.

Parametri

- `field`: Il campo da rimuovere dal documento. Può essere un campo singolo o un percorso punteggiato verso un campo nidificato.

Esempio (MongoDB Shell)

L'esempio seguente mostra come utilizzare l'`$unset`operatore per rimuovere il `Words` campo da un documento della raccolta. `example`

Crea documenti di esempio

```
db.example.insert({
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    },
    "Words": 2482
})
```

Esempio di interrogazione

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

Output

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

```
        "DoW": "Saturday"
    }
}
```

In questo esempio, l'`$unset`operatore viene utilizzato per rimuovere il `Words` campo dal documento `DocName` pari a «Documento 1". Il documento risultante non contiene più il `Words` campo.

Esempi di codice

Per visualizzare un esempio di codice per l'utilizzo del `$unset` comando, scegliete la scheda relativa alla lingua che desiderate utilizzare:

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()
```

Gestione di cluster e istanze Amazon DocumentDB

Queste sezioni riguardano i vari componenti e le attività correlate per la gestione dell'implementazione di Amazon DocumentDB (con compatibilità MongoDB).

Argomenti

- Panoramica delle attività operative di Amazon DocumentDB
- Panoramica dei cluster globali di Amazon DocumentDB
- Gestione dei cluster Amazon DocumentDB
- Gestione delle istanze di Amazon DocumentDB
- Gestione dei gruppi di sottoreti Amazon DocumentDB
- Disponibilità e replica elevate di Amazon DocumentDB
- Gestione degli eventi di Amazon DocumentDB
- Gestione dei gruppi di parametri del cluster Amazon DocumentDB
- Comprendere gli endpoint di Amazon DocumentDB
- Informazioni su Amazon DocumentDB Amazon Resource Names (ARN)
- Etichettatura delle risorse di Amazon DocumentDB
- Manutenzione di Amazon DocumentDB
- Comprendere i ruoli collegati ai servizi

Panoramica delle attività operative di Amazon DocumentDB

Questa sezione descrive le attività operative per il tuo cluster Amazon DocumentDB e come eseguire queste attività utilizzando. AWS CLI

Argomenti

- Aggiungere una replica a un cluster Amazon DocumentDB
- Descrizione di cluster e istanze
- Creazione di un'istantanea del cluster
- Ripristino da uno snapshot
- Rimuovere un'istanza da un cluster

- Eliminazione di un cluster

Aggiungere una replica a un cluster Amazon DocumentDB

Dopo aver creato l'istanza primaria per il tuo cluster Amazon DocumentDB, puoi aggiungere una o più repliche. Una replica è un'istanza di sola lettura che svolge due funzioni:

- Scalabilità: se hai un numero elevato di client che richiedono un accesso simultaneo, puoi aggiungere altre repliche per il ridimensionamento in lettura.
- Alta disponibilità: se l'istanza primaria fallisce, Amazon DocumentDB esegue automaticamente il failover su un'istanza di replica e la designa come nuova primaria. Se una replica non riesce, le altre istanze nel cluster saranno ancora in grado di assolvere le richieste finché non viene ripristinato il nodo con l'errore.

Ogni cluster Amazon DocumentDB può supportare fino a 15 repliche.

> **Note**
>
> Per la massima tolleranza ai guasti, devi distribuire le repliche in zone di disponibilità separate. Questo aiuta a garantire che il cluster Amazon DocumentDB possa continuare a funzionare, anche se un'intera zona di disponibilità diventa non disponibile.

L' AWS CLI esempio seguente mostra come aggiungere una nuova replica. Il parametro `--availability-zone` posiziona la replica nella zona di disponibilità specificata.

```
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
```

Descrizione di cluster e istanze

L' AWS CLI esempio seguente elenca tutti i cluster Amazon DocumentDB in una regione.
Per alcune funzionalità di gestione, come la gestione del ciclo di vita di cluster e istanze,

Amazon DocumentDB sfrutta la tecnologia operativa condivisa con Amazon RDS. Il parametro `filterName=engine,Values=docdb` filter restituisce solo i cluster Amazon DocumentDB.

Per ulteriori informazioni sulla descrizione e la modifica dei cluster, consulta Ciclo di vita del cluster Amazon DocumentDB.

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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",
            ...
        }
    ]
}
```

L' AWS CLI esempio seguente elenca le istanze in un cluster Amazon DocumentDB. Per ulteriori informazioni sulla descrizione e la modifica dei cluster, consulta Ciclo di vita delle istanze Amazon DocumentDB.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterMembers]'
```

L'aspetto dell'output è simile al seguente. In questo output, ci sono due istanze. L'istanza primaria è `sample-instance-1` (`"IsClusterWriter": true`). È presente anche un'istanza di replica, `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
            }
        ]
    ]
]
```

Creazione di un'istantanea del cluster

Uno snapshot del cluster è un backup completo dei dati nel cluster Amazon DocumentDB. Quando viene creata l'istantanea, Amazon DocumentDB legge i dati direttamente dal volume del cluster. Per questo motivo, puoi creare una snapshot anche se il cluster non contiene alcuna istanza attualmente in esecuzione. La quantità di tempo necessaria per creare una snapshot varia a seconda della dimensione del volume cluster.

Amazon DocumentDB supporta i backup automatici, che vengono eseguiti giornalmente durante la finestra di backup preferita, un periodo di 30 minuti durante il giorno. L' AWS CLI esempio seguente mostra come visualizzare la finestra di backup per il cluster:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].PreferredBackupWindow'
```

L'output mostra la finestra di backup (in UTC):

```
[
    "00:18-00:48"
]
```

Puoi definire la finestra di backup quando crei il tuo cluster Amazon DocumentDB. Puoi anche modificare la finestra di backup, come nell'esempio seguente: Se non definisci una finestra di backup, Amazon DocumentDB ne assegna automaticamente una al tuo cluster.

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --preferred-backup-window "02:00-02:30"
```

Oltre ai backup automatici, puoi creare manualmente una snapshot del cluster in qualsiasi momento. In questo caso, devi specificare il cluster di cui eseguire il backup e un nome univoco per la snapshot che consenta di ripristinarla in un secondo momento.

L' AWS CLI esempio seguente mostra come creare un'istantanea dei tuoi dati.

```
aws docdb create-db-cluster-snapshot \
    --db-cluster-identifier sample-cluster \
```

```
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Ripristino da uno snapshot

Puoi ripristinare un'istantanea del cluster in un nuovo cluster Amazon DocumentDB. A tale scopo, devi specificare il nome della snapshot e il nome di un nuovo cluster. Non è possibile eseguire il ripristino da un'istantanea a un cluster esistente; al contrario, Amazon DocumentDB crea un nuovo cluster al momento del ripristino e quindi lo compila con i dati dello snapshot.

L'esempio seguente mostra tutte le snapshot per il cluster `sample-cluster`.

```
aws docdb describe-db-cluster-snapshots \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusterSnapshots[*].[DBClusterSnapshotIdentifier,SnapshotType,Status]'
```

L'output è simile al seguente. Un'istantanea manuale viene creata manualmente, mentre un'istantanea automatica viene creata da Amazon DocumentDB all'interno della finestra di backup del cluster.

```
[
        "sample-cluster-snapshot",
        "manual",
        "available"
    ],
    [
        "rds:sample-cluster",
        "automated",
        "available"
    ]
]
```

L'esempio seguente mostra come ripristinare un cluster Amazon DocumentDB da un'istantanea.

```
aws docdb restore-db-cluster-from-snapshot \
    --engine docdb \
    --db-cluster-identifier new-sample-cluster \
    --snapshot-identifier sample-cluster-snapshot
```

Il nuovo cluster non ha istanze associate, quindi per interagire con questo cluster devi aggiungervi un'istanza:

```
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
```

È possibile utilizzare le seguenti AWS CLI operazioni per monitorare l'avanzamento della creazione di cluster e istanze. Quando lo stato del cluster e delle istanze è disponibile, puoi connetterti all'endpoint del nuovo cluster e accedere ai dati.

```
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]'
```

Rimuovere un'istanza da un cluster

Amazon DocumentDB archivia tutti i tuoi dati nel volume del cluster. I dati persistono nel volume cluster, anche se rimuovi tutte le istanze dal cluster. Se hai bisogno di accedere nuovamente ai dati, puoi aggiungere un'istanza al cluster in qualsiasi momento e riprendere da dove avevi lasciato.

L'esempio seguente mostra come rimuovere un'istanza dal cluster Amazon DocumentDB.

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-instance
```

Eliminazione di un cluster

Prima di poter eliminare un cluster Amazon DocumentDB, devi prima rimuovere tutte le sue istanze. L' AWS CLI esempio seguente restituisce informazioni sulle istanze di un cluster. Se questa operazione restituisce uno o più identificatori di istanza, devi eliminare tutte le istanze. Per ulteriori informazioni, consulta Rimuovere un'istanza da un cluster.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
```

```
    --query 'DBClusters[*].DBClusterMembers[*].DBInstanceIdentifier'
```

Dopo aver rimosso tutte le istanze, puoi eliminare il cluster. A questo punto, devi scegliere una delle seguenti opzioni:

- Crea un'istantanea finale: acquisisci tutti i dati del cluster in un'istantanea in modo da poter ricreare una nuova istanza con quei dati in un secondo momento. L'esempio seguente mostra la procedura per farlo:

```
aws docdb delete-db-cluster \
    --db-cluster-identifier sample-cluster \
    --final-db-snapshot-identifier sample-cluster-snapshot
```

- Salta l'istantanea finale: elimina definitivamente tutti i dati del cluster. Questa operazione non può essere annullata. L'esempio seguente mostra la procedura per farlo:

```
aws docdb delete-db-cluster \
    --db-cluster-identifier sample-cluster \
    --skip-final-snapshot
```

Panoramica dei cluster globali di Amazon DocumentDB

Cos'è un cluster globale?

Un cluster globale è composto da una regione principale e fino a 10 regioni secondarie di sola lettura. Le operazioni di scrittura vengono eseguite direttamente nel cluster primario nella regione primaria e Amazon DocumentDB replica automaticamente i dati nelle regioni secondarie utilizzando un'infrastruttura dedicata. La latenza è in genere inferiore a un secondo.

In che modo sono utili i cluster globali?

- Ripristino da Region-wide interruzioni: in caso di Region-wide interruzione, è possibile promuovere uno dei cluster secondari a cluster primario in pochi minuti, con un Recovery Time Objective (RTO) tipico inferiore a un minuto. Il Recovery Point Objective (RPO) viene in genere misurato in secondi, ma ciò dipende dal ritardo sulla rete al momento dell'errore.
- Letture globali con latenza locale: se hai uffici in tutto il mondo, puoi utilizzare un cluster globale per mantenere aggiornate le principali fonti di informazioni nella regione principale. Gli uffici nelle altre regioni possono accedere alle informazioni nella propria regione, con una latenza locale.

- Cluster secondari scalabili: puoi scalare i tuoi cluster secondari aggiungendo altre istanze di sola lettura a una regione secondaria. Il cluster secondario è di sola lettura, quindi può supportare fino a 16 istanze di replica di sola lettura anziché il solito limite di 15 per un singolo cluster.
- Replica rapida da cluster primari a cluster secondari: la replica eseguita da un cluster globale ha un impatto minimo sulle prestazioni sul cluster di database primario. Le risorse delle istanze database sono totalmente dedicate a servire carichi di lavoro di lettura e scrittura delle applicazioni.
- Flussi di modifica: è possibile leggere i flussi di modifica da un cluster secondario in cluster globali, riducendo il carico sul cluster primario e consentendo di scalare i carichi di lavoro di lettura dalle regioni secondarie.

Quali sono le attuali limitazioni dei cluster globali?

- I cluster globali non sono supportati su Amazon DocumentDB v3.6.
- I cluster globali sono supportati su tutti i tipi di istanza tranne db.t3, db.t4g e db.r4.
- I cluster globali non sono disponibili nelle seguenti regioni: Sud America (San Paolo), Europa (Milano), Cina (Pechino) e Cina (Ningxia).
- Lo switchover e il failover globale non sono supportati quando le regioni utilizzano versioni del motore diverse. Il failover manuale è supportato in caso di mancata corrispondenza tra le versioni del motore.
- Solo il cluster primario eseguire operazioni di scrittura. I client che eseguono operazioni di scrittura si connettono all'endpoint del cluster primario.
- Puoi avere un massimo di 10 regioni secondarie e una regione primaria per il tuo cluster.
- Un cluster secondario non può essere interrotto. Un cluster primario non può essere interrotto se è associato a cluster secondari. È possibile arrestare solo un cluster regionale che non ha cluster secondari.
- Le repliche collegate al cluster secondario possono riavviarsi in determinate circostanze. Se l'istanza della regione primaria si riavvia o fa il failover, vengono riavviate anche le repliche nella regione secondaria. Il cluster non è quindi disponibile finché tutte le repliche non saranno nuovamente sincronizzate con l'istanza di scrittura del cluster di database primario. Si tratta di un comportamento normale. Assicurati di aver compreso l'impatto sul cluster globale prima di apportare modifiche al cluster primario.

Argomenti

- Guida rapida: cluster globali

- Gestione di un cluster globale Amazon DocumentDB
- Connettiti a un cluster globale Amazon DocumentDB
- Monitoraggio dei cluster globali di Amazon DocumentDB
- Disaster recovery e cluster globali Amazon DocumentDB

Guida rapida: cluster globali

Argomenti

- Configurazione
- Creazione di un cluster globale Amazon DocumentDB
- Aggiungere un Regione AWS a un cluster globale Amazon DocumentDB
- Utilizzo di un'istantanea per il tuo cluster globale Amazon DocumentDB

Configurazione

Il cluster globale Amazon DocumentDB si estende su almeno due. Regioni AWS La regione primaria supporta un cluster con un'istanza primaria (writer) e fino a 15 istanze di replica, mentre una regione secondaria esegue un cluster di sola lettura composto interamente da un massimo di 16 istanze di replica. Un cluster globale può avere fino a 10 regioni secondarie. La tabella elenca il numero massimo di cluster, istanze e repliche consentite in un cluster globale.

Description	Primario Regione AWS	Secondario Regione AWS
Cluster	1	10 (massimo)
Istanze di scrittura	1	0
Read-only istanze (repliche Amazon DocumentDB), per cluster	15 (massimo)	16 (totali)
Read-only istanze (numero massimo consentito, dato il numero effettivo di regioni secondarie)	15 - s	s = numero totale di secondari e Regioni AWS

I cluster hanno i seguenti requisiti specifici:

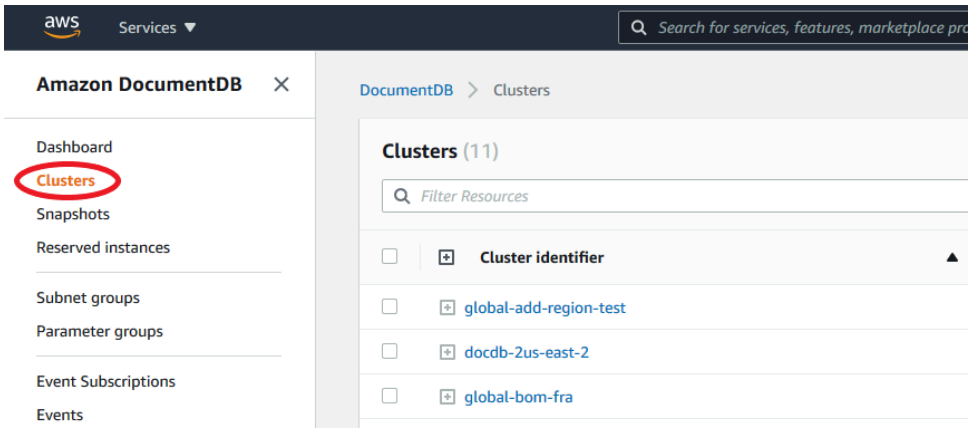
- Requisiti della classe dell'istanza del database: è possibile utilizzare solo le classi di `db.r6g` istanza `db.r5` e.
- Regione AWS requisiti: il cluster primario deve trovarsi in una regione e almeno un cluster secondario deve trovarsi in una regione diversa dello stesso account. È possibile creare fino a 10 cluster secondari (di sola lettura) e ognuno deve trovarsi in una regione diversa. In altre parole, non è possibile che due cluster si trovino nella stessa regione.
- Requisiti di denominazione: i nomi scelti per ciascuno dei cluster devono essere univoci in tutte le regioni. Non puoi usare lo stesso nome per cluster diversi anche se si trovano in regioni diverse.

Creazione di un cluster globale Amazon DocumentDB

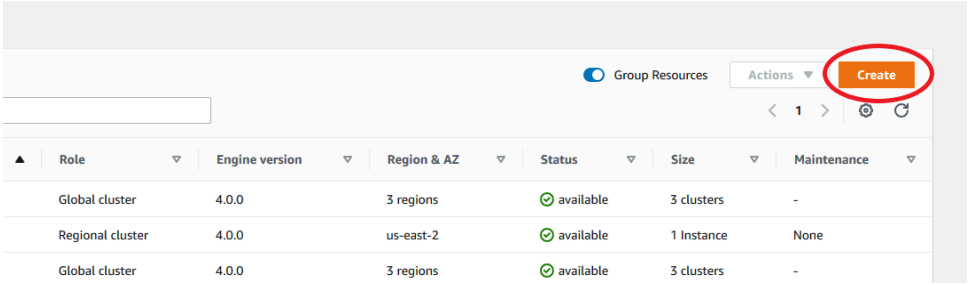
Sei pronto a creare il tuo primo cluster globale? In questa sezione spiegheremo come creare un nuovo cluster globale con nuovi cluster e istanze di database, utilizzando Console di gestione AWS o AWS CLI con le seguenti istruzioni.

Utilizzo di Console di gestione AWS

1. Nel Console di gestione AWS, accedi ad Amazon DocumentDB.
2. Quando accedi alla console Amazon DocumentDB, scegli Clusters.

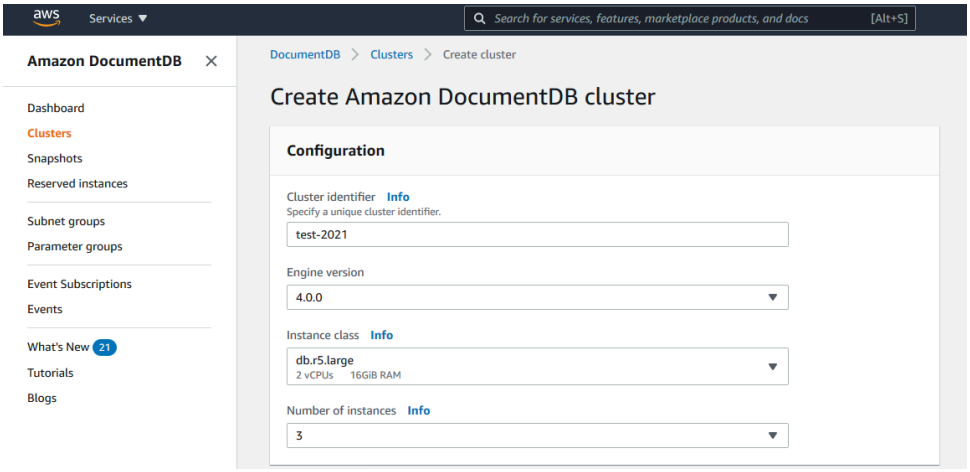


3. Scegli Create (Crea).

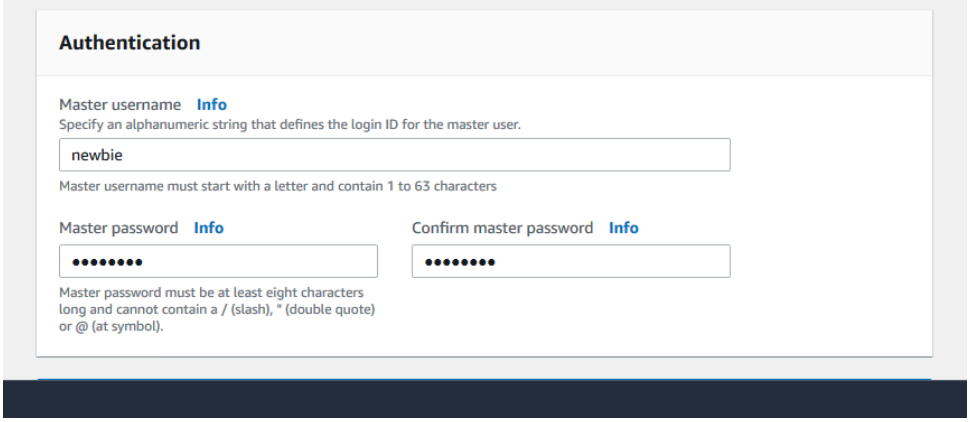


4. Compila di conseguenza la sezione di configurazione del modulo Create Amazon DocumentDB Cluster:

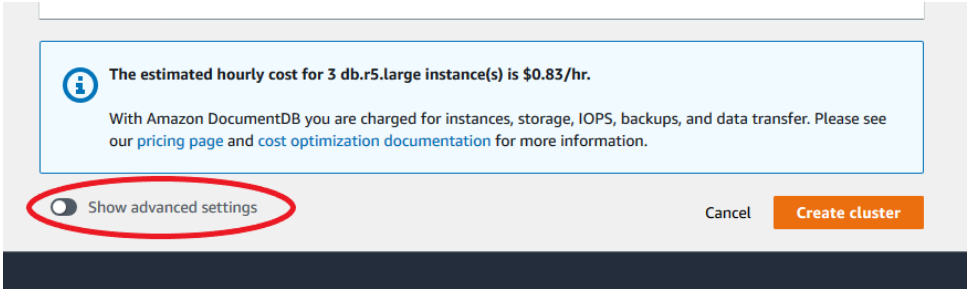
   - Identificatore del cluster: puoi inserire un identificatore univoco per questa istanza o consentire ad Amazon DocumentDB di fornire l'identificatore dell'istanza in base all'identificatore del cluster.
   - Versione del motore: scegli 4.0.0
   - Classe di istanza: scegli db.r5.large
   - Numero di istanze: scegli 3.



5. Nella sezione Autenticazione, inserisci un nome utente principale e una password principale.

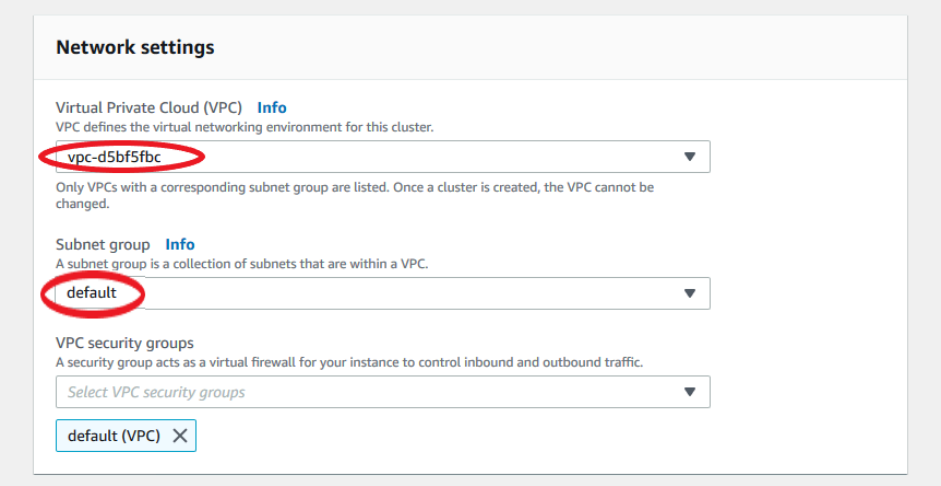


6. Scegli Mostra impostazioni avanzate.

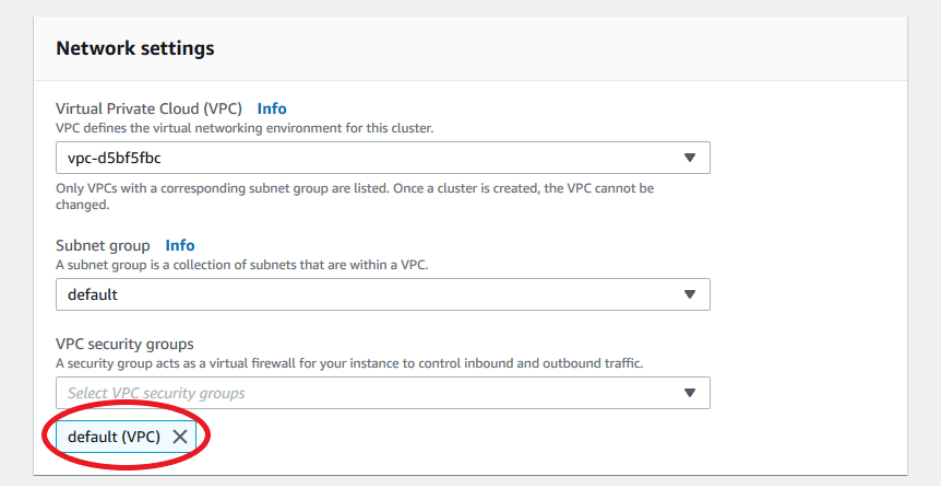


7. Nella sezione Impostazioni di rete:

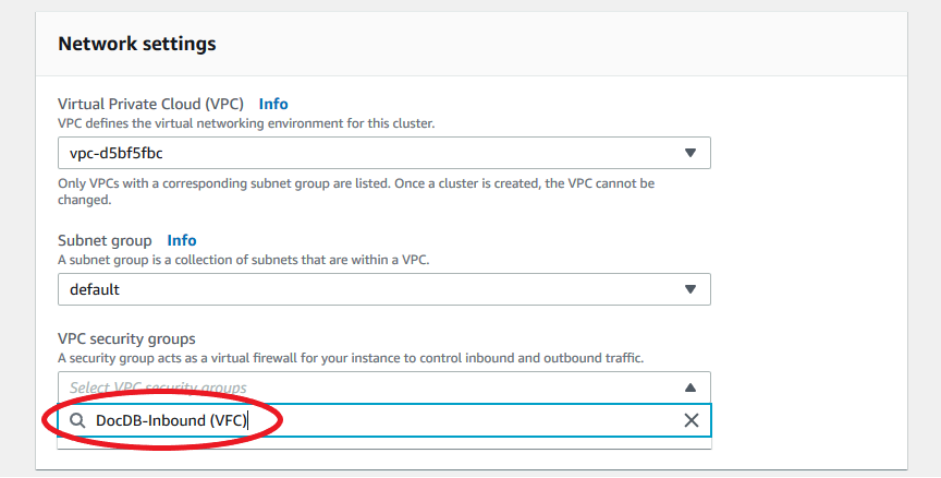
   - Mantieni le opzioni predefinite per Virtual Private Cloud (VPC) e Subnet group.



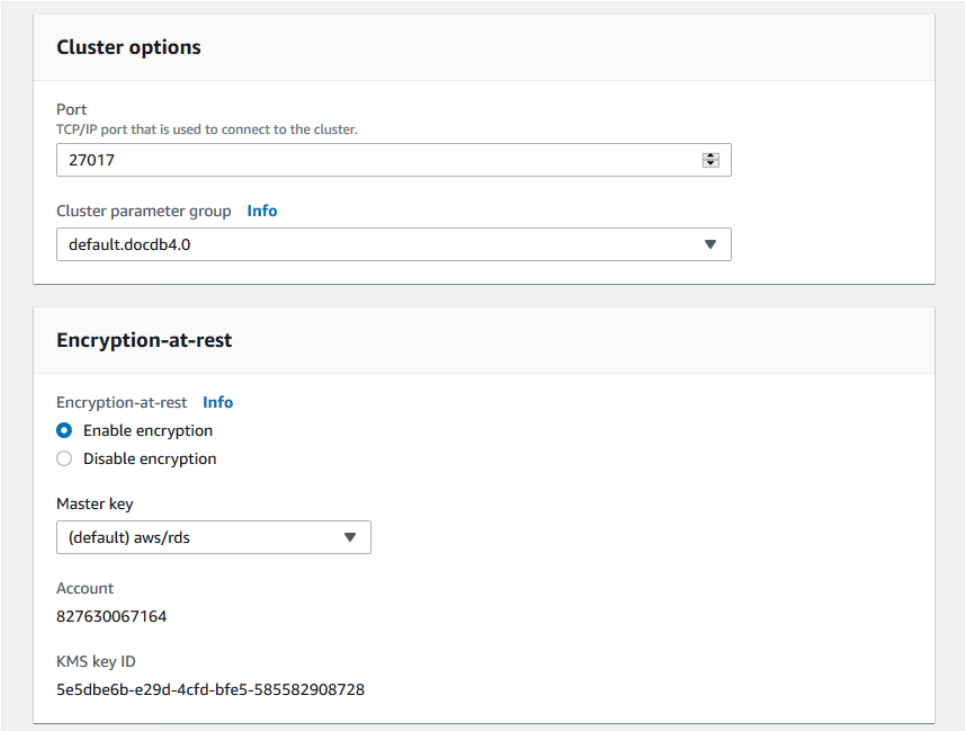
   - Per i gruppi di sicurezza VPC, l'impostazione predefinita (VPC) dovrebbe già essere aggiunta.



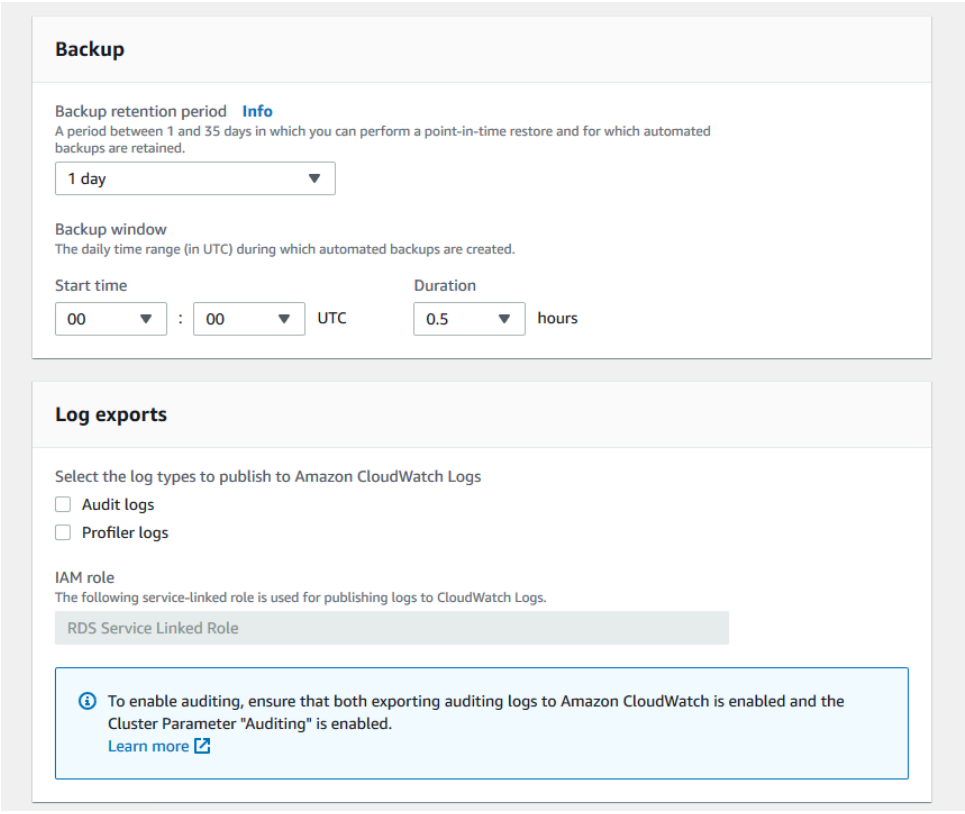
- Digita DocDB nel campo dei gruppi di sicurezza VPC e seleziona DocDB-Inbound (VFC).



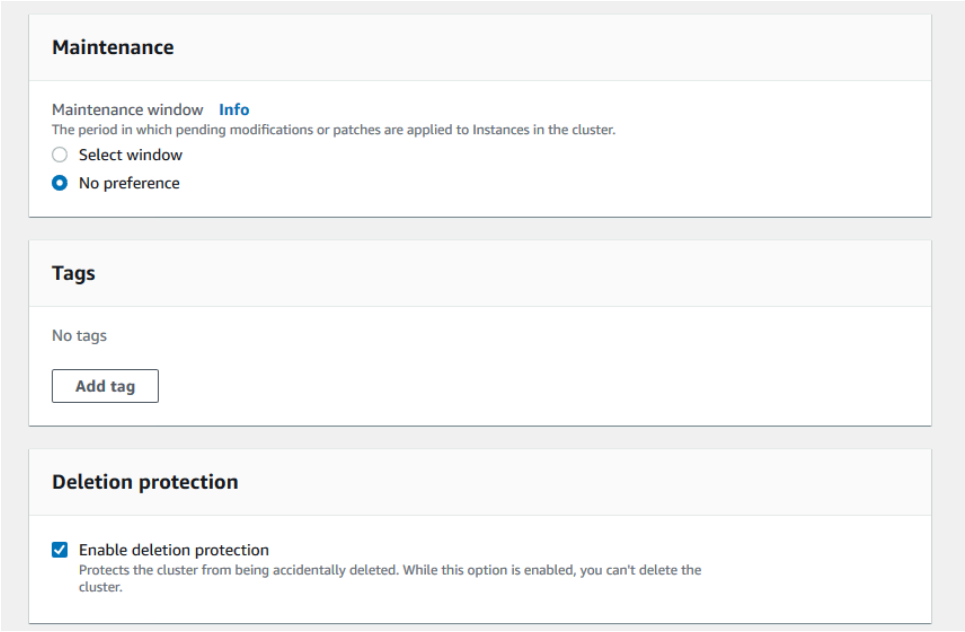
8. Per le opzioni del cluster e Encryption-at-rest, lascia le selezioni predefinite.



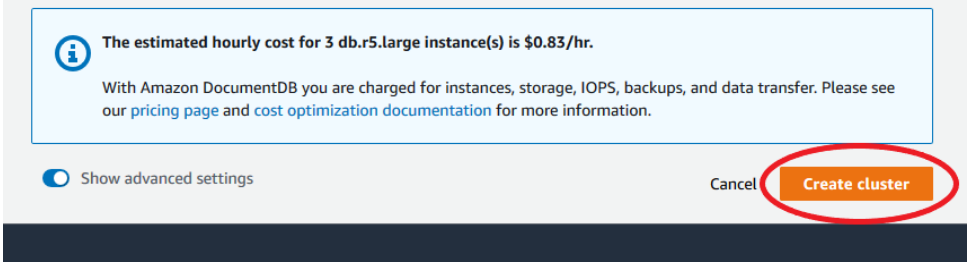
9. Per le esportazioni di backup e log, lasciate le selezioni predefinite.



10. Per la protezione da manutenzione , tag ed eliminazione, lasciate le selezioni predefinite.



11. Ora scegli Crea cluster.



Utilizzo di AWS CLI

Per creare un cluster regionale Amazon DocumentDB, chiama AWS CLI create-global-cluster. Il AWS CLI comando seguente crea un cluster Amazon DocumentDB denominato. `global-cluster-id` Per ulteriori informazioni sulla protezione dall'eliminazione, consulta. Eliminazione di un cluster Amazon DocumentDB

Inoltre, `--engine-version` è un parametro opzionale che per impostazione predefinita è l'ultima versione principale del motore. L'attuale versione predefinita del motore è `5.0.0` (nota: Amazon DocumentDB 8.0 è disponibile ma deve essere specificato esplicitamente come). `8.0.0` Quando vengono rilasciate nuove versioni principali del motore, la versione predefinita del motore per `--engine-version` viene aggiornata in modo da riflettere l'ultima versione principale del motore. Di conseguenza, per i carichi di lavoro di produzione, e in particolare quelli che dipendono da script, automazione o CloudFormation modelli, specificate esplicitamente la versione `--engine-version` principale prevista.

Se non `vpc-security-group-id` viene specificato un valore `db-subnet-group-name` o, Amazon DocumentDB utilizzerà il gruppo di sottorete e il gruppo di sicurezza Amazon VPC predefiniti per la regione specificata.

Nel seguente esempio, sostituisci ciascuno *`user input placeholder`* con le tue informazioni.

Per Linux, macOS o 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
```

Per 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
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "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"
    }
}
```

La creazione del cluster richiede diversi minuti. È possibile utilizzare Console di gestione AWS o AWS CLI per monitorare lo stato del cluster. Per ulteriori informazioni, consulta Monitoraggio dello stato di un cluster Amazon DocumentDB.

> **Important**
>
> Quando si utilizza il AWS CLI per creare un cluster regionale Amazon DocumentDB, non viene creata alcuna istanza. Di conseguenza, devi creare esplicitamente un'istanza primaria e le eventuali istanze di replica di cui hai bisogno. Puoi utilizzare la console o AWS CLI creare le istanze. Per ulteriori informazioni, consulta Aggiungere un'istanza Amazon DocumentDB a un cluster e crea DBCluster nell'API Reference di Amazon DocumentDB.

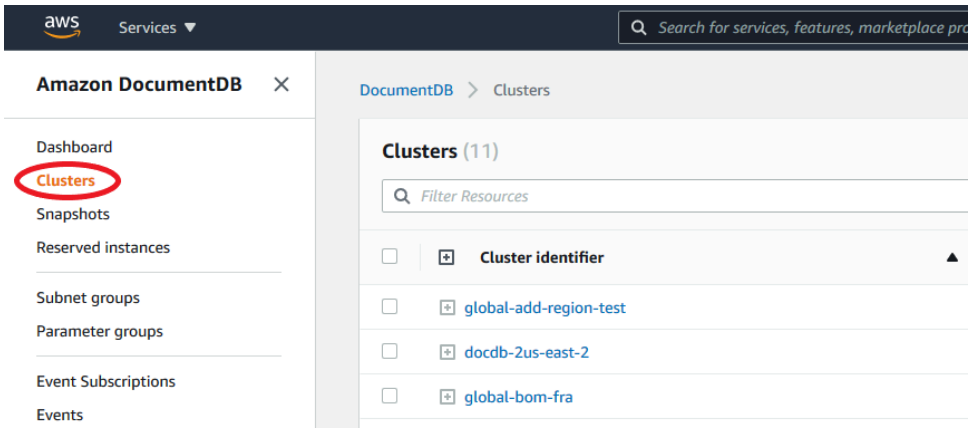
Una volta che il cluster regionale è disponibile, puoi aggiungere un cluster secondario in un'altra regione con le seguenti istruzioni:. Aggiungere un Regione AWS a un cluster globale Amazon DocumentDB Quando aggiungi una regione, il tuo cluster regionale diventa il tuo cluster principale, e hai un nuovo cluster secondario nella regione che hai scelto.

Aggiungere un Regione AWS a un cluster globale Amazon DocumentDB

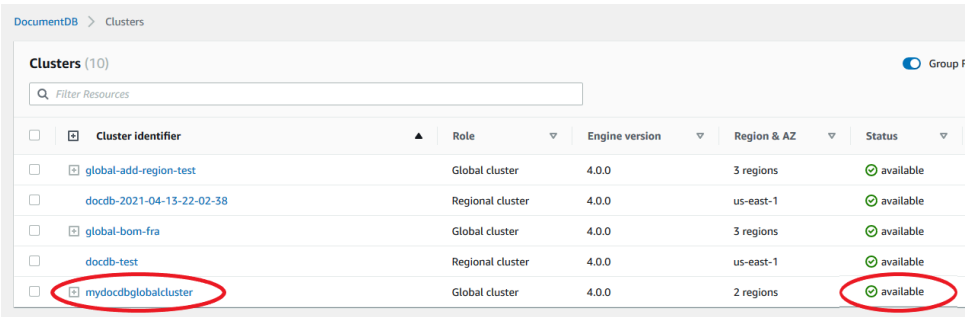
Un cluster globale richiede almeno un cluster secondario in una regione diversa rispetto al cluster primario ed è possibile aggiungere fino a 10 cluster secondari. Tieni presente che per ogni cluster secondario aggiunto, devi ridurre di una il numero di repliche consentite nel cluster primario. Ad esempio, se il cluster globale ha 10 regioni secondarie, il cluster primario può avere solo 5 (anziché 15) repliche. Per ulteriori informazioni, consulta Configurazione.

Utilizzo di Console di gestione AWS

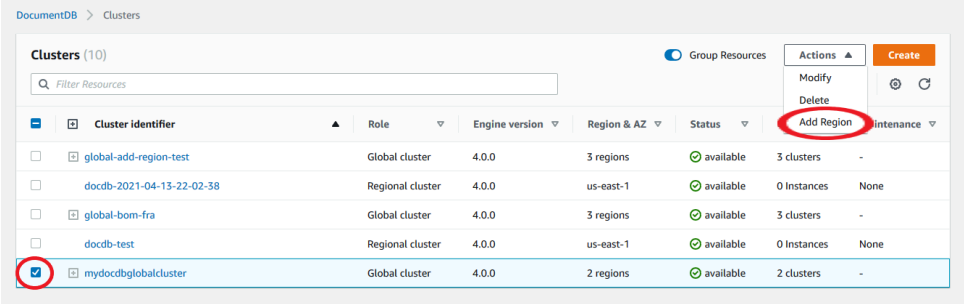
1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.
2. Nel pannello di navigazione scegliere Cluster.



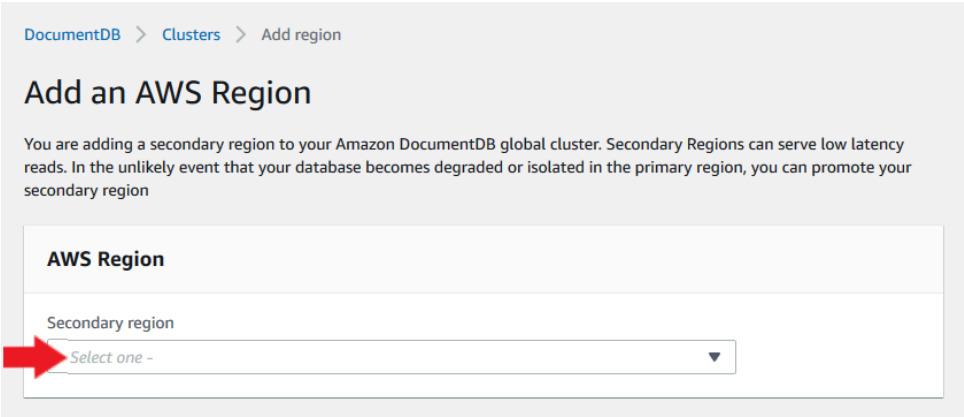
3. Scegli il cluster a cui desideri aggiungere un cluster secondario. Assicurati che il cluster lo sia`Available`.



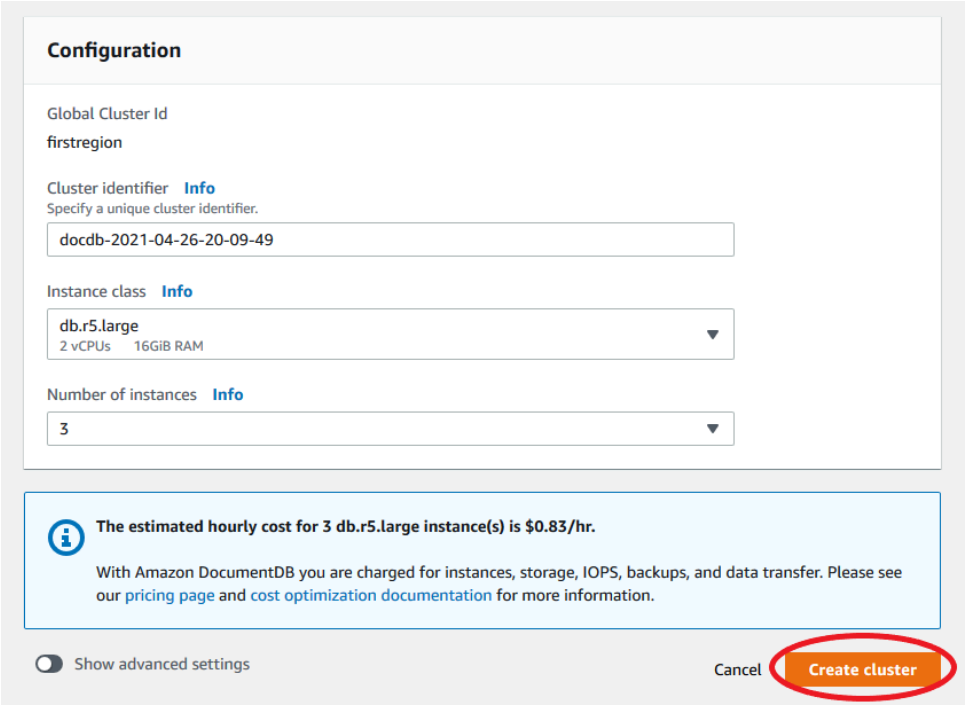
4. Seleziona l'elenco a discesa per Azioni, quindi scegli Aggiungi regione.



5. Nella Regione AWS pagina Aggiungi una, scegli la regione secondaria. Tieni presente che non puoi scegliere una regione che ha già un cluster secondario per lo stesso cluster globale. Inoltre, non può essere la stessa regione del cluster principale. Se questa è la prima regione che aggiungi, dovrai anche specificare un identificatore di cluster globale a tua scelta.



6. Completa i campi rimanenti per il cluster secondario nella nuova regione, quindi seleziona Crea cluster. Dopo aver aggiunto la regione, puoi visualizzarla nell'elenco dei cluster in. Console di gestione AWS



Utilizzo di AWS CLI

- Usa il comando `create-db-cluster` CLI con il nome (`--global-cluster-identifier`) del tuo cluster globale. Per gli altri parametri, effettuare le seguenti operazioni:
  - Per`--region`, scegli una regione Regione AWS diversa da quella della tua regione principale.
  - Scegli i valori specifici per i parametri `--engine` e `--engine-version`.
  - Per un cluster crittografato, specifica il tuo primario Regione AWS come cluster `--source-region` per la crittografia.

L'esempio seguente crea un nuovo cluster Amazon DocumentDB e lo collega al cluster globale come cluster secondario di sola lettura. Nell'ultimo passaggio, l'istanza viene aggiunta al nuovo cluster.

Nell'esempio seguente, sostituisci ciascuno *`user input placeholder`* con le tue informazioni.

Per Linux, macOS o 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
```

Per 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
```

Utilizzo di un'istantanea per il tuo cluster globale Amazon DocumentDB

Puoi ripristinare un'istantanea di un cluster Amazon DocumentDB da utilizzare come punto di partenza per il tuo cluster globale. A tale scopo, è necessario ripristinare l'istantanea e creare un nuovo cluster. Questo fungerà da cluster principale del cluster globale. È quindi possibile aggiungere un'altra regione al cluster ripristinato, convertendolo così in un cluster globale.

Gestione di un cluster globale Amazon DocumentDB

La maggior parte delle operazioni di gestione viene eseguita sui singoli cluster che compongono un cluster globale. Quando scegli Risorse relative al gruppo nella pagina Cluster della console, vengono visualizzati il cluster primario e i cluster secondari raggruppati nel cluster globale associato.

La scheda Configurazione per un cluster globale mostra il Regioni AWS luogo in cui i cluster sono in esecuzione, la versione e l'identificatore globale del cluster.

Argomenti

- Modifica di un cluster globale Amazon DocumentDB
- Modifica dei parametri di un cluster globale Amazon DocumentDB
- Rimuovere un cluster da un cluster globale Amazon DocumentDB
- Eliminazione di un cluster da un cluster globale Amazon DocumentDB
- Creazione di un cluster Amazon DocumentDB headless in una regione secondaria

Modifica di un cluster globale Amazon DocumentDB

La pagina Cluster Console di gestione AWS elenca tutti i cluster globali, mostrando il cluster primario e i cluster secondari per ciascuno di essi. Il cluster globale ha le proprie impostazioni di configurazione. In particolare, ha delle regioni associate ai suoi cluster primari e secondari.

Quando apporti modifiche al cluster globale, hai la possibilità di annullare le modifiche.

Quando si sceglie Continue (Continua), si confermano le modifiche.

Modifica dei parametri di un cluster globale Amazon DocumentDB

È possibile configurare i gruppi di parametri del cluster in modo indipendente per ogni cluster all'interno del cluster globale. La maggior parte dei parametri funziona come per altri tipi di cluster Amazon DocumentDB. Mantieni le impostazioni coerenti tra tutti i cluster di un database globale per evitare cambiamenti di comportamento imprevisti se promuovi un cluster secondario come primario.

Ad esempio, utilizzare le stesse impostazioni per fusi orari e set di caratteri per evitare comportamenti incoerenti se un cluster diverso diventa un cluster primario.

Rimuovere un cluster da un cluster globale Amazon DocumentDB

Esistono diverse situazioni in cui potresti voler rimuovere i cluster dal tuo cluster globale. Ad esempio, potresti voler rimuovere un cluster da un cluster globale se il cluster primario diventa degradato o isolato. Diventa quindi un cluster con provisioning autonomo che può essere utilizzato per creare un nuovo cluster globale. Per ulteriori informazioni, consulta Esecuzione di un failover manuale per un cluster globale Amazon DocumentDB.

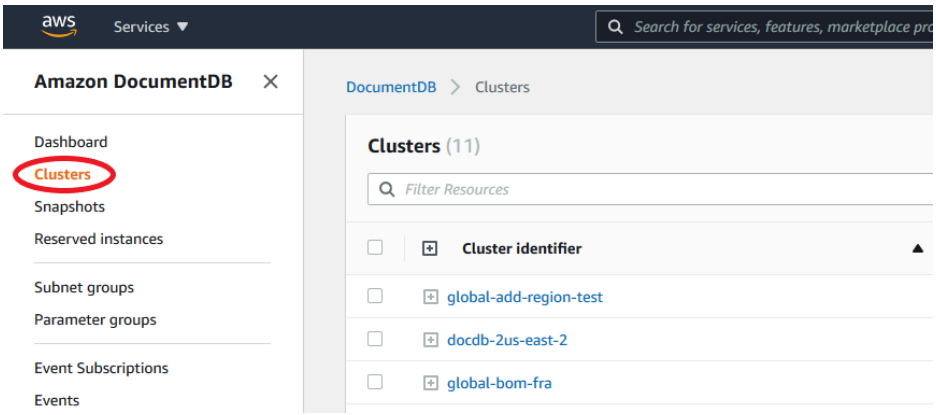
Potresti voler rimuovere i cluster anche perché desideri eliminare un cluster globale che non ti serve più. Non puoi eliminare il cluster globale se non dopo aver scollegato tutti i cluster associati, lasciando il primario per ultimo. Per ulteriori informazioni, consulta Eliminazione di un cluster da un cluster globale Amazon DocumentDB.

> **Note**
>
> Quando un cluster è scollegato dal cluster globale, non è più sincronizzato con quello primario. Diventa un cluster con provisioning autonomo con funzionalità complete. read/write Inoltre, non è più visibile nella console Amazon DocumentDB. È visibile solo quando si seleziona la regione nella console in cui si trovava il cluster.

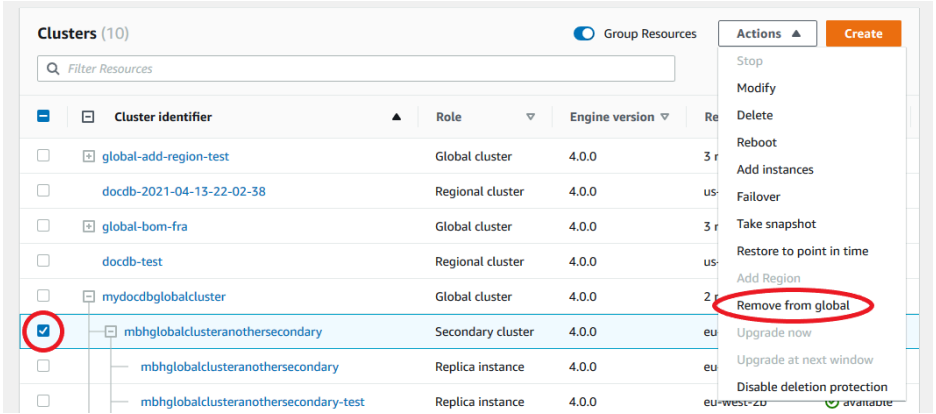
Puoi rimuovere i cluster dal tuo cluster globale utilizzando l' Console di gestione AWS API AWS CLI, the o RDS.

Using the Console di gestione AWS

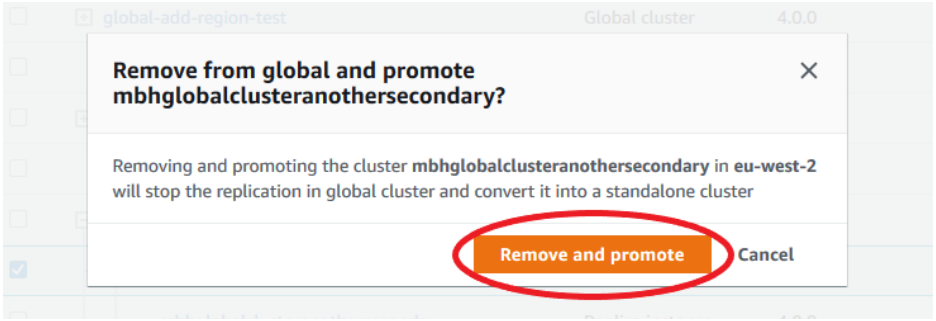
1. Accedi Console di gestione AWS e accedi alla console Amazon DocumentDB.
2. Scegli Clusters nella barra di navigazione a sinistra.



3. Espandi il cluster globale in modo da poter vedere tutti i cluster secondari. Seleziona i cluster secondari che desideri rimuovere. Scegli Azioni e, nel menu a discesa, scegli Rimuovi da globale.



4. Apparirà un prompt che ti chiederà di confermare che desideri scollegare il cluster secondario dal cluster globale. Scegli Rimuovi e promuovi per rimuovere il cluster dal cluster globale.



Ora quel cluster non funge più da secondario e non è più sincronizzato con il cluster primario. È un cluster autonomo con funzionalità complete read/write .

Dopo aver rimosso o eliminato tutti i cluster secondari, è possibile rimuovere il cluster primario nello stesso modo. È possibile scollegare o rimuovere il cluster primario dal cluster globale solo dopo aver rimosso tutti i cluster secondari. Il cluster globale potrebbe rimanere nell'elenco dei cluster, con zero regioni e AZ. Puoi eliminare se non desideri più utilizzare questo cluster globale.

Using the AWS CLI

Per rimuovere un cluster da un cluster globale, esegui il comando `remove-from-global-cluster` CLI con i seguenti parametri:

- `--global-cluster-identifier`— Il nome (identificatore) del cluster globale.
- `--db-cluster-identifier`— Il nome di ogni cluster da rimuovere dal cluster globale.

Gli esempi seguenti rimuovono prima un cluster secondario e poi il cluster primario da un cluster globale.

Per Linux, macOS o 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
```

Ripeti il `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` comando per ogni regione secondaria del cluster globale.

Per 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
```

Ripeti il `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` comando per ogni regione secondaria del cluster globale.

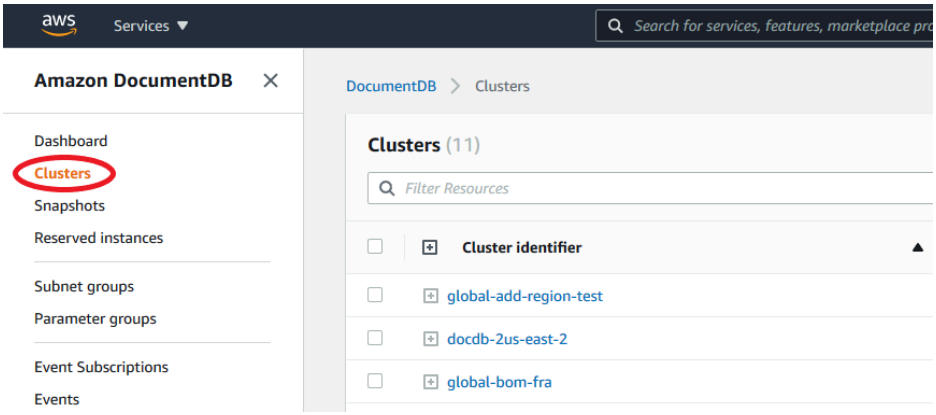
Eliminazione di un cluster da un cluster globale Amazon DocumentDB

Per eliminare un cluster globale, procedi come segue:

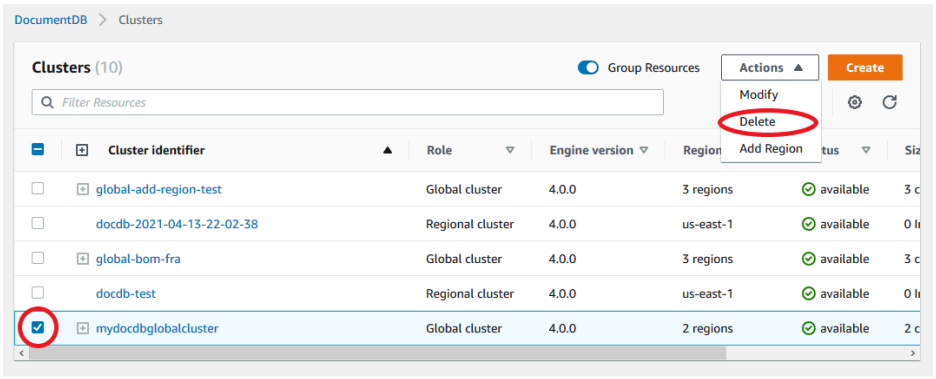
- Rimuovi tutti i cluster secondari dal cluster globale. Ogni cluster diventa un cluster autonomo. Vedi la sezione precedente,Rimuovere un cluster da un cluster globale Amazon DocumentDB.
- Da ogni cluster autonomo, elimina tutte le repliche.
- Rimuovi il cluster primario dal cluster globale. Questo diventa un cluster autonomo.
- Dal cluster primario, elimina prima tutte le repliche, quindi elimina l'istanza primaria. L'eliminazione dell'istanza primaria dal nuovo cluster autonomo in genere rimuove anche sia il cluster che il cluster globale.

Using the Console di gestione AWS

1. Accedi Console di gestione AWS e accedi alla console Amazon DocumentDB.
2. Scegli Clusters e trova il cluster globale che desideri eliminare.



3. Con il cluster globale selezionato, scegli Elimina dal menu Azioni.



Conferma che tutti i cluster siano stati rimossi dal cluster globale. Il cluster globale dovrebbe mostrare zero regioni e AZ e una dimensione pari a zero cluster. Se il cluster globale contiene dei cluster, non puoi ancora eliminarlo. Dovrai prima seguire le istruzioni nel passaggio precedente, Rimuovere un cluster da un cluster globale Amazon DocumentDB.

Using the AWS CLI

Per eliminare un cluster globale, esegui il comando `delete-global-cluster` CLI con il nome Regione AWS e l'identificatore globale del cluster, come mostrato nell'esempio seguente.

Per Linux, macOS o Unix:

```
aws docdb --region primary_region delete-global-cluster \
   --global-cluster-identifier global_cluster_id
```

Per Windows:

```
aws docdb --region primary_region delete-global-cluster ^
   --global-cluster-identifier global_cluster_id
```

Creazione di un cluster Amazon DocumentDB headless in una regione secondaria

Sebbene un cluster globale Amazon DocumentDB richieda almeno un cluster secondario Regione AWS diverso da quello primario, puoi utilizzare una configurazione headless per il cluster secondario. Un cluster Amazon DocumentDB secondario headless è un cluster Amazon DocumentDB privo di istanza. Questo tipo di configurazione può ridurre le spese per un cluster globale. In un cluster Amazon DocumentDB, elaborazione e storage sono disaccoppiati. Senza l'istanza, non ti verrà addebitato alcun costo per l'elaborazione, ma solo per lo storage. Se è impostato correttamente, il volume di archiviazione di un headless secondario viene mantenuto sincronizzato con il cluster primario.

Aggiungi il cluster secondario come fai normalmente quando crei un cluster globale Amazon DocumentDB. Tuttavia, dopo che il cluster primario ha iniziato la replica su quello secondario, si elimina l'istanza di sola lettura dal cluster secondario. Questo cluster secondario è ora considerato «headless» perché non ha più un'istanza. Tuttavia, il volume di storage viene mantenuto sincronizzato con il cluster Amazon DocumentDB primario.

> **Important**
>
> Usa i cluster headless solo se riesci a tollerare Region-wide guasti per 15 minuti o più. Il ripristino da un Region-wide errore con un cluster secondario headless richiede la creazione di una nuova istanza dopo il failover. Una nuova istanza può richiedere 10-15 minuti per diventare disponibile.

Come aggiungere un cluster secondario headless al cluster globale

1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.
2. Scegli Clusters nella barra di navigazione a sinistra.
3. Scegli il cluster globale che necessita di un cluster secondario. Assicurati che il cluster primario lo sia`Available`.
4. Per Azioni, scegli Aggiungi regione.
5. Nella pagina Aggiungi una regione, scegli la regione secondaria.

> **Note**
>
> Non puoi scegliere una regione che ha già un cluster secondario per lo stesso cluster globale. Inoltre, non può essere la stessa regione del cluster principale.

6. Completa i campi rimanenti per il cluster secondario nella nuova regione . Queste sono le stesse opzioni di configurazione di qualsiasi istanza del cluster.
7. Aggiungi una regione. Dopo aver aggiunto la regione al cluster globale, questa viene visualizzata `Clusters` nell'elenco di Console di gestione AWS.
8. Controlla lo stato del cluster secondario e della relativa istanza del lettore prima di continuare, utilizzando Console di gestione AWS o il AWS CLI. Ecco un comando di esempio se utilizzi AWS CLI:

```
$ aws docdb describe-db-clusters --db-cluster-identifier secondary-cluster-id --
query '*[].[Status]' --output text
```

   Possono essere necessari diversi minuti prima che lo stato di un cluster secondario appena aggiunto passi da creazione a disponibile. Quando il cluster è disponibile, puoi eliminare l'istanza del lettore.
9. Seleziona l'istanza del lettore nel cluster secondario, quindi scegli Elimina.
10. Dopo aver eliminato l'istanza del lettore, il cluster secondario rimane parte del cluster globale. Non dovrebbe avere alcuna istanza associata.

> **Note**
>
> Puoi utilizzare questo cluster Amazon DocumentDB secondario headless per ripristinare manualmente il tuo cluster globale Amazon DocumentDB da un'interruzione non pianificata nella regione primaria, se si verifica tale interruzione.

Connettiti a un cluster globale Amazon DocumentDB

Il modo in cui ci si connette a un cluster globale dipende dalla necessità di scrivere sul cluster o di leggere dal cluster:

- Per richieste o interrogazioni di sola lettura, ti connetti all'endpoint di lettura del cluster nel tuo. Regione AWS
- Per eseguire le istruzioni DML (Data Manipulation Language) o DDL (Data Definition Language), esegui la connessione all'endpoint del cluster per il cluster primario. Questo endpoint potrebbe trovarsi in un'applicazione diversa. Regione AWS

Quando visualizzi un cluster globale nella console, puoi vedere tutti gli endpoint generici associati a tutti i relativi cluster.

Il modo in cui ci si connette a un cluster globale dipende dalla necessità di scrivere nel database o di leggere dal database. Per le operazioni DDL, DML e di lettura che desideri eseguire dalla regione principale, connettiti al cluster primario utilizzando l'endpoint del cluster in modalità set di repliche, con una preferenza di lettura di. `secondaryPreferred=true` Questo indirizza il traffico di scrittura all'istanza di scrittura del cluster primario e il traffico di lettura all'istanza di replica del cluster primario.

Per il traffico interregionale di sola lettura, connettiti a uno dei cluster secondari utilizzando l'endpoint del cluster in modalità replica set. Poiché tutte le istanze sono istanze di replica di sola lettura, specifica una preferenza di lettura diversa `primary` da (ad esempio,, o). `secondary` `secondaryPreferred` `nearest` Per ridurre al minimo la latenza, scegliere qualsiasi endpoint di lettura che si trova nella propria regione o nella regione più vicina.

Monitoraggio dei cluster globali di Amazon DocumentDB

Amazon DocumentDB (con compatibilità con MongoDB) si integra CloudWatch in modo da poter raccogliere e analizzare le metriche operative per i tuoi cluster. Puoi monitorare queste metriche utilizzando la CloudWatch console, la console Amazon DocumentDB, la () o l'API. AWS Command Line Interface AWS CLI CloudWatch

Per monitorare un cluster globale, utilizza le seguenti CloudWatch metriche.

Metrica	Description
`GlobalClusterReplicatedWriteIO`	Il numero medio di I/O operazioni di scrittura fatturate replicate dal volume del cluster nel volume primario Regione AWS a quello del cluster in un volume secondario Regione AWS, segnalato a intervalli di 5 minuti. Il numero

Metrica	Description
	di repliche in ciascuna regione secondaria è uguale `ReplicatedWriteIOs` al numero di repliche nella regione interna `VolumeWriteIOPs` eseguite dalla regione primaria.
`GlobalClusterDataTransferBytes`	La quantità di dati trasferiti dal cluster primario Regione AWS a quello di un cluster secondario si misura in Regione AWS byte.
`GlobalClusterReplicationLag`	La quantità di ritardo, in millisecondi, durante la replica degli eventi di modifica dal cluster primario a quello di un cluster secondario Regione AWS Regione AWS

Per ulteriori informazioni su come visualizzare queste metriche, consulta Visualizzazione dei dati. CloudWatch

Disaster recovery e cluster globali Amazon DocumentDB

Argomenti

- Esecuzione di un failover gestito per un cluster globale Amazon DocumentDB
- Esecuzione di un failover manuale per un cluster globale Amazon DocumentDB
- Esecuzione di uno switchover per un cluster globale Amazon DocumentDB
- Sblocco dello switchover o del failover di un cluster globale
- Gestione degli RPO per i cluster globali di Amazon DocumentDB

Utilizzando un cluster globale, puoi recuperare rapidamente da disastri come guasti regionali. Il ripristino da emergenza viene in genere misurato utilizzando i valori per RTO e RPO.

- Obiettivo del tempo di ripristino (RTO): il tempo necessario a un sistema per tornare in uno stato funzionante dopo un'emergenza. In altre parole, l'RTO misura i tempi di inattività. Per un cluster globale, RTO in pochi minuti.
- Obiettivo del punto di ripristino (RPO): la quantità di dati che possono essere persi (misurata nel tempo). Per un cluster globale, l'RPO viene in genere misurato in secondi.

- Per ripristinare il sistema da un'interruzione non pianificata, è possibile eseguire un failover interregionale su uno dei sistemi secondari del cluster globale. Quando il tuo cluster globale ha più regioni secondarie, assicurati di scollegare tutte le regioni secondarie che desideri promuovere come principali. Quindi, promuovi una di quelle regioni secondarie come nuova primaria. Regione AWS Infine, crei nuovi cluster in ciascuna delle altre regioni secondarie e li colleghi al tuo cluster globale.

Esecuzione di un failover gestito per un cluster globale Amazon DocumentDB

Questo approccio è destinato alla continuità aziendale in caso di una reale emergenza a livello regionale o di un'interruzione completa del livello di servizio.

Durante un failover gestito, il cluster primario viene trasferito nella regione secondaria prescelta mentre viene mantenuta la topologia di replica esistente del cluster globale Amazon DocumentDB. Il cluster secondario scelto promuove uno dei suoi nodi di sola lettura allo stato di istanza di scrittura completa. Questo passaggio consente al cluster di assumere il ruolo di cluster primario. Il database non sarà disponibile per un breve periodo di tempo mentre il cluster sta assumendo il suo nuovo ruolo. I dati che non sono stati replicati dal vecchio cluster primario al cluster secondario scelto potrebbero mancare quando questo secondario diventa il nuovo primario. Il vecchio volume primario fa del suo meglio per scattare un'istantanea prima di sincronizzarsi con il nuovo primario, in modo da preservare i dati non replicati nell'istantanea.

> **Note**
>
> È possibile eseguire un failover gestito tra più regioni su un cluster globale Amazon DocumentDB solo se il cluster primario e tutti i cluster secondari hanno le stesse versioni del motore. Se le versioni del motore non sono compatibili, puoi eseguire il failover manualmente seguendo i passaggi indicati in Esecuzione di un failover manuale per un cluster globale Amazon DocumentDB.
>
> Se le versioni del motore della regione non corrispondono, il failover verrà bloccato. Verifica eventuali aggiornamenti in sospeso e applicali per assicurarti che tutte le versioni del motore della regione corrispondano e che il failover globale del cluster sia sbloccato. Per ulteriori informazioni, consulta Sblocco dello switchover o del failover di un cluster globale.

Per ridurre al minimo la perdita di dati, procedi come segue prima di utilizzare questa funzionalità:

- Porta le applicazioni offline per evitare che le scritture vengano inviate al cluster primario del cluster globale Amazon DocumentDB.
- Controlla i tempi di ritardo per tutti i cluster secondari di Amazon DocumentDB. La scelta della regione secondaria con il minor ritardo di replica può ridurre al minimo la perdita di dati relativamente all'attuale regione primaria in stato di errore. Verifica i tempi di ritardo per tutti i cluster secondari di Amazon DocumentDB nel cluster globale visualizzando la metrica in Amazon. `GlobalClusterReplicationLag` CloudWatch Queste metriche mostrano il ritardo (in millisecondi) della replica su un cluster secondario rispetto al cluster primario.

  Per ulteriori informazioni sulle CloudWatch metriche per Amazon DocumentDB, consulta. Metriche di Amazon DocumentDB

Durante un failover gestito, il cluster secondario scelto viene promosso al nuovo ruolo di primario. Tuttavia, non eredita le varie opzioni di configurazione del cluster primario. Una mancata corrispondenza nella configurazione può causare problemi di prestazioni, incompatibilità dei carichi di lavoro e altri comportamenti anomali. Per evitare tali problemi, risolvi le differenze tra i cluster globali di Amazon DocumentDB per quanto segue:

- Configura un gruppo di parametri del cluster Amazon DocumentDB per il nuovo primario, se necessario: puoi configurare i gruppi di parametri del cluster Amazon DocumentDB in modo indipendente per ogni cluster nel tuo cluster globale Amazon DocumentDB. Pertanto, quando promuovi un cluster secondario affinché assuma il ruolo principale, il gruppo di parametri del secondario potrebbe essere configurato in modo diverso rispetto a quello primario. In tal caso, modifica il gruppo di parametri del cluster secondario promosso in modo che sia conforme alle impostazioni del cluster primario. Per scoprire come, consulta Modifica dei gruppi di parametri del cluster Amazon DocumentDB.
- Configura gli strumenti e le opzioni di monitoraggio, come CloudWatch gli eventi e gli allarmi di Amazon: configura il cluster promosso con la stessa capacità di registrazione, gli stessi allarmi e così via necessari per il cluster globale. Come per i gruppi di parametri, la configurazione di queste funzionalità non viene ereditata dal primario durante il processo di failover. Alcune CloudWatch metriche, come il ritardo di replica, sono disponibili solo per le regioni secondarie. Pertanto, un failover modifica il modo in cui visualizzare tali metriche e impostare i relativi allarmi e potrebbe richiedere modifiche da apportare a qualsiasi dashboard predefinito. Per ulteriori informazioni sui cluster e sul monitoraggio di Amazon DocumentDB, consulta. Monitoraggio e registrazione in Amazon DocumentDB

In genere, il cluster secondario scelto assume il ruolo principale entro un minuto. Non appena il nodo di scrittura della nuova regione primaria è disponibile, puoi connettervi le tue applicazioni e riprendere i tuoi carichi di lavoro. Dopo che Amazon DocumentDB ha promosso il nuovo cluster primario, ricostruisce automaticamente tutti i cluster regionali secondari aggiuntivi.

Poiché i cluster globali di Amazon DocumentDB utilizzano la replica asincrona, il ritardo di replica in ciascuna regione secondaria può variare. Amazon DocumentDB ricostruisce queste regioni secondarie in modo da avere esattamente gli stessi dati point-in-time del nuovo cluster di regioni primarie. La durata dell'attività di ricostruzione completa può richiedere da alcuni minuti a diverse ore, a seconda delle dimensioni del volume di archiviazione e della distanza tra regioni. Quando i cluster regionali secondari terminano la ricostruzione in base alla nuova regione primaria, diventano disponibili per l'accesso in lettura. Non appena il nuovo writer primario viene promosso e disponibile, il cluster della nuova regione primaria può gestire le operazioni di lettura e scrittura per il cluster globale Amazon DocumentDB.

Per ripristinare la topologia originale del cluster globale, Amazon DocumentDB monitora la disponibilità della vecchia regione primaria. Non appena tale regione è integra e nuovamente disponibile, Amazon DocumentDB la aggiunge automaticamente al cluster globale come regione secondaria. Prima di creare il nuovo volume di storage nella vecchia regione primaria, Amazon DocumentDB tenta di scattare un'istantanea del vecchio volume di storage nel punto in cui si è verificato l'errore. Ciò consente di usare lo snapshot per recuperare i dati mancanti. Se questa operazione ha esito positivo, Amazon DocumentDB inserisce questa istantanea denominata «rds:docdb-unplanned-global-failover-name-of-old-primary-» nella sezione snapshot del. DB-cluster-timestamp Console di gestione AWS Puoi anche vedere questa istantanea `DescribeDBClusterSnapshots` elencata nelle informazioni restituite dall'operazione API.

> **Note**
>
> Lo snapshot del vecchio volume di archiviazione è uno snapshot del sistema soggetto al periodo di conservazione del backup configurato sul vecchio cluster primario. Per conservare questo snapshot oltre il periodo di conservazione, puoi copiarlo e salvarlo come snapshot manuale. Per ulteriori informazioni sulla copia degli snapshot, inclusi i prezzi, consulta Copiare un'istantanea del cluster.

Dopo il ripristino della topologia originale, è possibile riportare il cluster globale alla regione primaria originale eseguendo un'operazione di switchover nel momento più opportuno per l'azienda e il carico

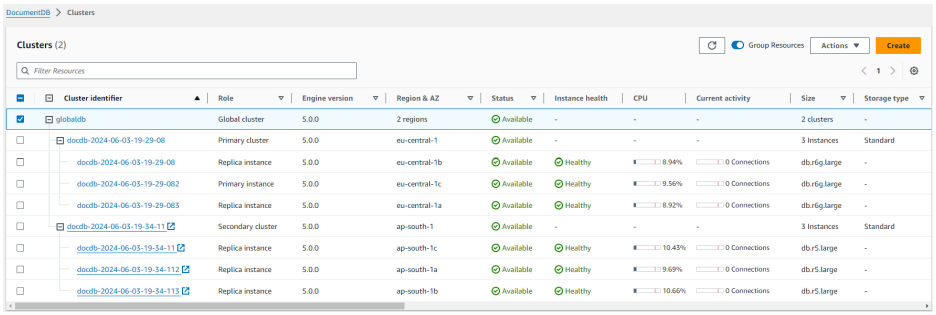
di lavoro. A tale scopo, segui la procedura in [Esecuzione di uno switchover per un cluster globale Amazon DocumentDB].

Puoi eseguire il failover del tuo cluster globale Amazon DocumentDB utilizzando l'API Console di gestione AWS AWS CLI, the o Amazon DocumentDB.

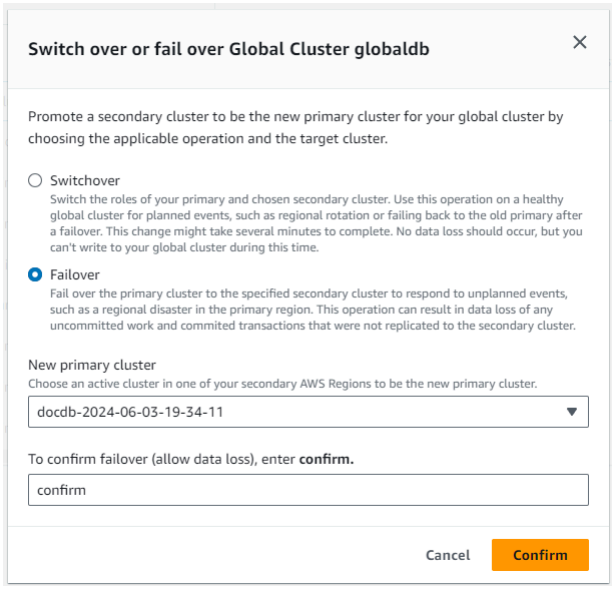
Using the Console di gestione AWS

Esegui un failover gestito sul tuo cluster globale Amazon DocumentDB

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.
3. Trova e scegli il cluster globale Amazon DocumentDB su cui desideri eseguire il failover.



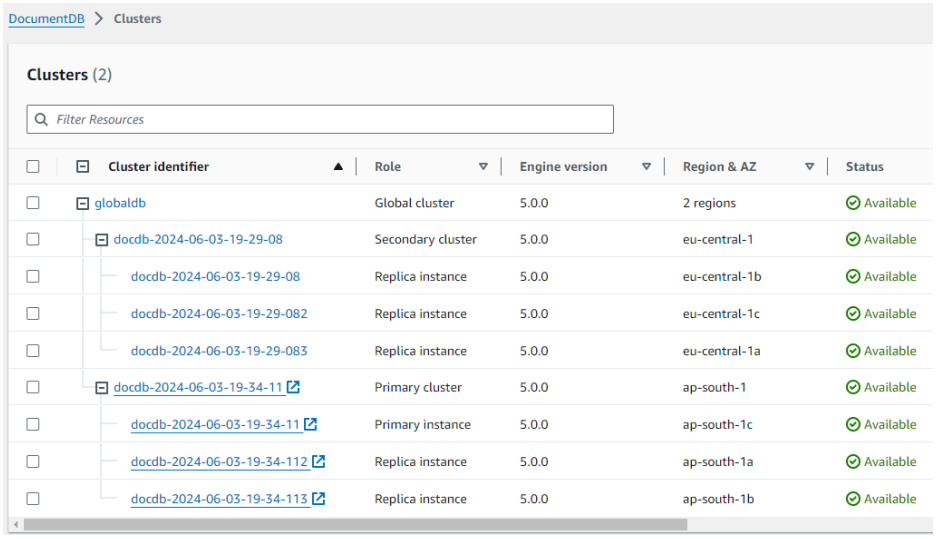
4. Scegli Switchover o Failover dal menu Azioni.
5. Nella finestra di dialogo visualizzata, scegli Failover, quindi scegli il cluster secondario dall'elenco a discesa Nuovo cluster primario.



6. Digita «conferma» nell'ultimo campo. Quindi scegli Conferma.

Lo stato del cluster primario cambia in "Failing-over». Questa condizione dovrebbe richiedere circa un minuto. Durante questo periodo, lo stato del nuovo cluster primario mostra "Modifica in corso... ». Una volta promosso, il nuovo primario mostrerà "Disponibile" e sarà in grado di fornire transazioni di lettura e scrittura. Le regioni secondarie, inclusa la vecchia primaria, mostreranno "Risincronizzazione... «mentre si risincronizza con la nuova primaria. Analogamente al nuovo primario, sarà in grado di gestire la transazione solo quando lo stato sarà impostato su "Disponibile».

7. Una volta completato, il cluster primario originale diventa il cluster secondario. Il cluster secondario selezionato diventa il cluster primario.



Using the AWS CLI

Esegui un failover gestito sul tuo cluster globale Amazon DocumentDB

Esegui il comando `failover-global-cluster` CLI per eseguire il failover del tuo cluster globale Amazon DocumentDB. Con il comando, passa i valori per le seguenti opzioni:

- `--region`
- `--global-cluster-identifier`
- `--target-db-cluster-identifier`
- `--allow-data-loss`

Nei seguenti esempi, sostituisci ciascuno *`user input placeholder`* con le informazioni del tuo cluster.

Per Linux, macOS o 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
```

Per 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
```

Esecuzione di un failover manuale per un cluster globale Amazon DocumentDB

Se un intero cluster in un cluster Regione AWS diventa non disponibile, puoi promuovere la funzionalità di un altro cluster nel cluster globale. read/write

È possibile attivare manualmente il meccanismo di failover globale del cluster se un cluster in un altro Regione AWS è la scelta migliore per fungere da cluster principale. Ad esempio, potrebbe essere necessario incrementare la capacità di uno dei cluster secondari e quindi promuoverlo a cluster primario. Oppure l'equilibrio delle attività tra di essi Regioni AWS potrebbe cambiare, quindi il passaggio dal cluster primario a un altro Regione AWS potrebbe comportare una latenza inferiore per le operazioni di scrittura.

La procedura seguente illustra cosa fare per promuovere uno dei cluster secondari in un cluster globale Amazon DocumentDB.

Per promuovere un cluster secondario:

1. Interrompi l'emissione di istruzioni DML e altre operazioni di scrittura sul cluster primario in caso Regione AWS di interruzione.
2. Identifica un cluster da uno secondario da Regione AWS utilizzare come nuovo cluster primario. Se ne hai due (o più) secondari Regioni AWS nel tuo cluster globale, scegli il cluster secondario con il minor tempo di ritardo.
3. Scollega il cluster secondario scelto dal cluster globale.

La rimozione di un cluster secondario da un cluster globale interrompe immediatamente la replica dal primario a questo secondario e la promuove a cluster con provisioning autonomo con funzionalità complete. read/write Qualsiasi altro cluster secondario associato al cluster primario nella regione interessata dall'interruzione è ancora disponibile e può accettare chiamate dall'applicazione. Inoltre consumano risorse. Poiché stai ricreando il cluster globale, per evitare split-brain e altri problemi, rimuovi gli altri cluster secondari prima di creare il nuovo cluster globale nei passaggi seguenti.

Per i passaggi dettagliati per lo scollegamento, consulta Rimuovere un cluster da un cluster globale Amazon DocumentDB.

4. Questo cluster diventa il cluster principale di un nuovo cluster globale quando inizi ad aggiungervi regioni, nella fase successiva.
5. Aggiungi un Regione AWS al cluster. Quando esegui questa operazione, inizia il processo di replica da primario a secondario.
6. Aggiungine altre Regioni AWS se necessario per ricreare la topologia necessaria a supportare l'applicazione. Assicurati che le scritture delle applicazioni vengano inviate al cluster corretto prima, durante e dopo aver apportato modifiche come queste, per evitare incoerenze tra i dati tra i cluster del cluster globale (problemi di split-brain).
7. Quando l'interruzione è stata risolta e sei pronto per assegnare nuovamente il cluster originale Regione AWS come cluster primario, esegui gli stessi passaggi in senso inverso.
8. Rimuovi uno dei cluster secondari dal cluster globale. Ciò gli consentirà di servire il read/write traffico.
9. Reindirizza tutto il traffico di scrittura al cluster primario dell'originale Regione AWS.
10. Aggiungi un Regione AWS per configurare uno o più cluster secondari nello stesso modo di prima Regione AWS .

I cluster globali di Amazon DocumentDB possono essere gestiti tramite AWS SDK, consentendo di creare soluzioni per automatizzare il processo di failover globale dei cluster per i casi d'uso di Disaster Recovery e Business Continuity Planning. Una di queste soluzioni è resa disponibile per i nostri clienti con licenza Apache 2.0 ed è accessibile dal nostro archivio di strumenti qui. https://github.com/awslabs/amazon-documentdb-tools/tree/master/global-clusters-automation Questa soluzione sfrutta Amazon Route 53 per la gestione degli endpoint e fornisce AWS Lambda funzioni che possono essere attivate in base a eventi appropriati.

Esecuzione di uno switchover per un cluster globale Amazon DocumentDB

Utilizzando gli switchover, è possibile modificare regolarmente la Regione del cluster primario. Questo approccio è destinato agli scenari controllati, ad esempio durante la manutenzione operativa e altre procedure operative pianificate.

Esistono tre casi d'uso comuni per l'utilizzo degli switchover:

- Per i requisiti relativi alla "rotazione regionale" imposti a settori specifici. Ad esempio, le normative sui servizi finanziari potrebbero imporre che i sistemi di livello 0 passino a un'altra regione per diversi mesi per garantire l'esecuzione regolare delle procedure di ripristino di emergenza.
- Per applicazioni con approccio "follow-the-sun" in più regioni. Ad esempio, un'azienda potrebbe voler fornire scritture con latenza inferiore in diverse regioni in base all'orario di lavoro nei vari fusi orari.
- Come metodo senza perdita di dati per eseguire il failback alla regione principale originale dopo un failover.

> **Note**
>
> Gli switchover sono progettati per essere utilizzati su un cluster globale Amazon DocumentDB sano. Per eseguire il ripristino da un'interruzione non pianificata, puoi eseguire la procedura appropriata descritta in Esecuzione di un failover manuale per un cluster globale Amazon DocumentDB.
>
> Per eseguire uno switchover, tutte le regioni secondarie devono eseguire esattamente la stessa versione del motore di quella principale. Se le versioni del motore della regione non corrispondono, il passaggio verrà bloccato. Verifica eventuali aggiornamenti in sospeso e applicali per assicurarti che tutte le versioni del motore della regione corrispondano e che il passaggio al cluster globale sia sbloccato. Per ulteriori informazioni, consulta Sblocco dello switchover o del failover di un cluster globale.

Durante un passaggio, Amazon DocumentDB commuta il cluster primario nella regione secondaria prescelta mantenendo la topologia di replica esistente del cluster globale. Prima di avviare il processo di passaggio, Amazon DocumentDB attende che tutti i cluster regionali secondari siano completamente sincronizzati con il cluster di regione principale. Il cluster database nella regione primaria diventa di sola lettura e il cluster secondario scelto promuove uno dei relativi nodi di sola lettura allo stato di nodo di scrittura completa. La promozione di questo nodo a nodo di scrittura

consente a tale cluster secondario di assumere il ruolo di cluster primario. Poiché tutti i cluster secondari sono stati sincronizzati con il primario all'inizio del processo, il nuovo primario continua le operazioni per il cluster globale Amazon DocumentDB senza perdere alcun dato. Il database non è disponibile per un breve periodo, mentre i cluster primario e secondario selezionati assumono i loro nuovi ruoli.

Per ottimizzare la disponibilità delle applicazioni, procedi come segue prima di utilizzare questa funzionalità:

- Eseguite questa operazione durante le ore non di punta o in un altro momento in cui le scritture sul cluster primario sono minime.
- Porta le applicazioni offline per evitare che le scritture vengano inviate al cluster primario del cluster globale Amazon DocumentDB.
- Controlla i tempi di ritardo per tutti i cluster secondari di Amazon DocumentDB nel cluster globale visualizzando la metrica in Amazon. `GlobalClusterReplicationLag` CloudWatch Questa metrica mostra quanto è indietro (in millisecondi) la replica su un cluster secondario rispetto al cluster primario. Questo valore è direttamente proporzionale al tempo impiegato da Amazon DocumentDB per completare il passaggio. Di conseguenza, maggiore è il valore del ritardo, maggiore sarà il tempo necessario per lo switchover.

  Per ulteriori informazioni sulle CloudWatch metriche per Amazon DocumentDB, consulta. Metriche di Amazon DocumentDB

Durante uno switchover gestito, il cluster database secondario scelto viene promosso al nuovo ruolo primario. Tuttavia, non eredita le varie opzioni di configurazione del cluster di database primario. Una mancata corrispondenza nella configurazione può causare problemi di prestazioni, incompatibilità dei carichi di lavoro e altri comportamenti anomali. Per evitare tali problemi, risolvi le differenze tra i cluster globali di Amazon DocumentDB per quanto segue:

- Configura il gruppo di parametri del cluster Amazon DocumentDB DB per il nuovo primario, se necessario: puoi configurare i gruppi di parametri del cluster Amazon DocumentDB in modo indipendente per ogni cluster nel tuo cluster globale Amazon DocumentDB. Ciò significa che quando si promuove un cluster di database secondario perché assuma il ruolo primario, il gruppo di parametri dal secondario potrebbe essere configurato in modo diverso rispetto al primario. In tal caso, modifica il gruppo di parametri del cluster di database secondario promosso in modo che sia conforme alle impostazioni del cluster primario. Per scoprire come, consulta Gestione dei gruppi di parametri del cluster Amazon DocumentDB.

- Configura strumenti e opzioni di monitoraggio, come Amazon CloudWatch Events and alarms: configura il cluster promosso con la stessa capacità di registrazione, gli stessi allarmi e così via necessari per il cluster globale. Come per i gruppi di parametri, la configurazione di queste funzionalità non viene ereditata dal ruolo primario durante il processo di switchover. Alcune CloudWatch metriche, come il ritardo di replica, sono disponibili solo per le regioni primarie. Pertanto, uno switchover modifica il modo in cui visualizzare tali metriche e impostare i relativi allarmi e potrebbe richiedere modifiche da apportare a qualsiasi dashboard predefinito. Per ulteriori informazioni, consulta Monitoraggio e registrazione in Amazon DocumentDB.

> **Note**
>
> In genere, lo switchover del ruolo può richiedere fino a diversi minuti.

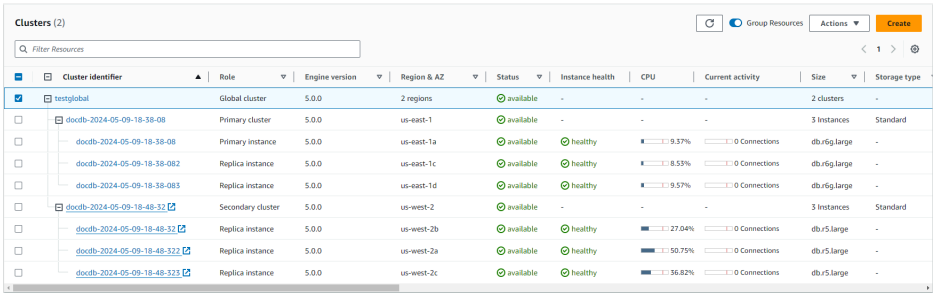
Una volta completato il processo di passaggio, il cluster Amazon DocumentDB promosso può gestire le operazioni di scrittura per il cluster globale.

Puoi passare al tuo cluster globale Amazon DocumentDB utilizzando o: Console di gestione AWS AWS CLI

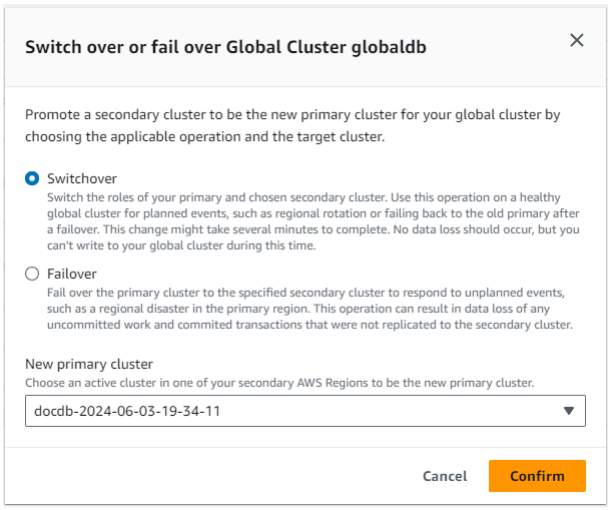
Using the Console di gestione AWS

Esegui uno switchover sul tuo cluster globale Amazon DocumentDB

1. Accedi a e apri la Console di gestione AWS console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.
3. Trova e seleziona il cluster globale Amazon DocumentDB a cui desideri passare.



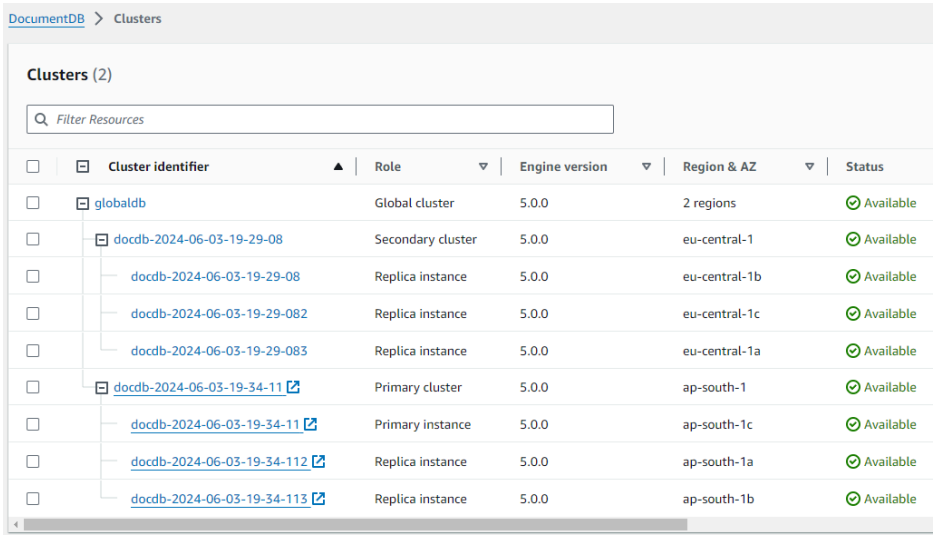
4. Scegli Switchover o Failover dal menu Azioni.
5. Nella finestra di dialogo visualizzata, scegli Switchover, quindi scegli il cluster secondario dall'elenco a discesa Nuovo cluster primario.



6. Scegli Conferma.

   Lo stato del cluster primario cambia in "Switching-over». Questa condizione dovrebbe richiedere circa tre minuti. Durante questo periodo, lo stato di tutti i cluster regionali mostra "Modifica in corso... ». Una volta sincronizzate le regioni e promossa la nuova primaria, verrà visualizzato "Disponibile" per tutti i campi di stato e sarà in grado di gestire le transazioni.

7. Una volta completato, il cluster primario originale diventa il cluster secondario. Il cluster secondario selezionato diventa il cluster primario.



Using the AWS CLI

Esegui uno switchover sul tuo cluster globale Amazon DocumentDB

Esegui il comando `switchover-global-cluster` CLI per passare al cluster globale Amazon DocumentDB. Con il comando, passa i valori per le seguenti opzioni:

- `--region`
- `--global-cluster-identifier`
- `--target-db-cluster-identifier`

Nei seguenti esempi, sostituisci ciascuno *`user input placeholder`* con le informazioni del tuo cluster.

Per Linux, macOS o 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
```

Per 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
```

Sblocco dello switchover o del failover di un cluster globale

Regioni con diverse versioni del motore

Gli switchover e i failover dei cluster globali vengono bloccati quando non tutti i cluster regionali del cluster globale utilizzano la stessa versione del motore. Se le versioni non corrispondono, potresti visualizzare questo errore quando chiami uno switchover o un failover: il cluster DB di destinazione specificato sta eseguendo una versione del motore con un livello di patch diverso rispetto al cluster DB di origine. Applica regolarmente le versioni più recenti del motore per mantenere integri i cluster globali.

Per risolvere questo errore, aggiorna prima tutte le regioni secondarie e poi la regione principale alla stessa versione del motore applicando eventuali azioni di manutenzione in sospeso. Per visualizzare

le azioni di manutenzione in sospeso e applicare le modifiche necessarie per correggere il problema, segui le istruzioni in una delle seguenti schede:

Using the Console di gestione AWS

Per sbloccare uno switchover o un failover globale di un cluster, devi determinare se ci sono azioni di manutenzione in sospeso per i tuoi cluster e applicarle. Segui questi passaggi per visualizzare e applicare le azioni di manutenzione:

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.
2. Nel pannello di navigazione scegliere Cluster.
3. Nella tabella Clusters, individua il cluster globale nella colonna Identificatore del cluster. Nel cluster globale, prendi nota di ogni cluster secondario e del cluster primario per il cluster globale specificato ed esegui i passaggi seguenti per ciascuno.
4. Per ogni cluster secondario:
    a. Se è disponibile un aggiornamento per il cluster, viene indicato come Disponibile , Richiesto o Finestra successiva nella colonna Manutenzione.
    b. Per eseguire un'azione, scegli il cluster per visualizzarne i dettagli, quindi scegli Manutenzione e backup. Vengono visualizzati gli elementi di manutenzione in sospeso.
    c. In Descrizione, se indica che è disponibile un «Nuovo aggiornamento di manutenzione», selezionalo e scegli Applica ora.
5. Per il tuo cluster principale:
    a. Se è disponibile un aggiornamento per il cluster, viene indicato come Disponibile , Richiesto o Finestra successiva nella colonna Manutenzione.
    b. Per eseguire un'azione, scegli il cluster per visualizzarne i dettagli, quindi scegli Manutenzione e backup. Vengono visualizzati gli elementi di manutenzione in sospeso.
    c. In Descrizione, se indica che è disponibile un «Nuovo aggiornamento di manutenzione», selezionalo e scegli Applica ora.

Using the AWS CLI

Per sbloccare uno switchover o un failover globale del cluster, devi determinare se ci sono azioni di manutenzione in sospeso per il cluster e applicarle. Segui questi passaggi per visualizzare e

applicare le azioni di manutenzione prima sui cluster secondari e poi sul cluster primario del tuo cluster globale:

1. Esegui quanto segue prima sul cluster regionale di ciascuna regione secondaria e poi per il cluster regionale primario delle regioni.
2. Esegui il comando `describe-pending-maintenance-actions` CLI con l'`--resource-identifier`opzione per determinare se sono disponibili azioni di manutenzione per il tuo cluster regionale Amazon DocumentDB.

   Nei seguenti esempi, sostituisci ciascuno *`user input placeholder`* con le informazioni del tuo cluster.

   Per Linux, macOS o Unix:

   ```
   aws docdb describe-pending-maintenance-action \
      --resource-identifier arn:aws:rds:us-
   east-1:001234567890:cluster:docdb-2025-03-27-19-21-15
   ```

   Per Windows:

   ```
   aws docdb describe-pending-maintenance-action ^
      --resource-identifier arn:aws:rds:us-
   east-1:001234567890:cluster:docdb-2025-03-27-19-21-15
   ```

   Il risultato è simile a questo:

   ```
   {
       "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. Se è necessaria un'azione di manutenzione, esegui il comando `apply-pending-maintenance-action` CLI con le seguenti opzioni:

   - `--resource-identifier`
   - `--apply-action`
   - `--opt-in-type`
   - `--region`

   Nei seguenti esempi, sostituisci ciascuno *`user input placeholder`* con le informazioni del tuo cluster.

   Per Linux, macOS o 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
```

   Per 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. Una volta completata l'azione di manutenzione, esegui nuovamente il `describe-pending-maintenance-actions` comando per assicurarti che non vi siano altre azioni in sospeso per il cluster.

   Il risultato desiderato è:

```
{
    "PendingMaintenanceActions": []
}
```

Using the Amazon DocumentDB API

Per sbloccare uno switchover o un failover globale del cluster, è necessario determinare se sono presenti azioni di manutenzione in sospeso per il cluster e applicarle. Utilizza le seguenti API per visualizzare e applicare le azioni di manutenzione:

1. Esegui quanto segue prima sul cluster regionale di ciascuna regione secondaria e poi per il cluster regionale primario delle regioni.
2. Chiama l'PendingMaintenanceActionAPI per determinare se sono disponibili azioni di manutenzione per il tuo cluster globale Amazon DocumentDB.
3. Applica eventuali modifiche chiamando l'ApplyPendingMaintenanceActionAPI.

Modifiche in sospeso sul cluster di destinazione

Gli switchover e i failover vengono bloccati anche quando il cluster secondario di destinazione ha una modifica pianificata che non è stata ancora applicata, ad esempio un'azione di modifica o manutenzione richiesta per la successiva finestra di manutenzione. In questo caso, potresti visualizzare questo errore quando chiami uno switchover o un failover: non puoi eseguire il failover sul cluster con ARN *arn:aws:rds:us-east-1:001234567890:cluster:docdb-2025-03-27-19-21-15* perché è in fase di modifica o perché le modifiche sono in sospeso. Riprova quando il cluster sarà disponibile. Per sbloccare l'operazione, rimuovi la modifica pianificata sul cluster di destinazione. I passaggi dipendono dal tipo di modifica. Una volta rimossa la modifica pianificata e ripristinato lo stato del cluster`available`, riprova lo switchover o il failover.

Annulla un'azione di manutenzione programmata

Non puoi annullare un'azione di manutenzione programmata (annullare un opt-in) da Console di gestione AWS, quindi usa il. AWS CLI

Using the AWS CLI

1. Esegui il comando CLI describe-pending-maintenance-actions con l'opzione e cerca un'azione di manutenzione il `--resource-identifier` cui nome sia. `OptInStatus` `next-maintenance`

   Negli esempi seguenti, sostituisci ciascuno con le informazioni del tuo cluster. *`user input placeholder`*

   Per Linux, macOS o 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
   ```

   Per 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
   ```

   Il risultato è simile al seguente. Nota il `Action` valore (`os-upgrade`in questo esempio), che utilizzerai nel passaggio successivo.

   ```
   {
       "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. Annulla l'azione pianificata eseguendo il comando CLI apply-pending-maintenance-action con`--opt-in-type undo-opt-in`, passando il valore del passaggio precedente a. `Action` `--apply-action`

   Per Linux, macOS o 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
```

   Per 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
```

   Nella risposta, l'azione di manutenzione non ha più un, che conferma che l'opt-in pianificato è stato `OptInStatus` annullato.

```
{
    "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"
            }
        ]
    }
}
```

Annulla una modifica pianificata del cluster

Using the AWS CLI

1. Verifica le modifiche pianificate eseguendo il comando CLI describe-db-clusters e controllando il campo di output. `PendingModifiedValues`

   > **Note**
   >
   > Usa la CLI di Amazon RDS (`aws rds`) per questo comando, non la CLI di Amazon DocumentDB (), perché l'API Amazon DocumentDB non restituisce un campo a livello di `aws docdb` cluster. `describe-db-clusters PendingModifiedValues`

   Nei seguenti esempi, sostituisci ciascuno con le informazioni del tuo cluster. *`user input placeholder`*

   Per Linux, macOS o Unix:

   ```
   aws rds describe-db-clusters \
      --db-cluster-identifier docdb-2025-03-27-19-21-15 \
      --query 'DBClusters[0].PendingModifiedValues' \
      --region us-east-1
   ```

   Per Windows:

   ```
   aws rds describe-db-clusters ^
      --db-cluster-identifier docdb-2025-03-27-19-21-15 ^
      --query "DBClusters[0].PendingModifiedValues" ^
      --region us-east-1
   ```

   Il risultato mostra le modifiche pianificate. In questo esempio, è in sospeso una modifica al nome del cluster (`DBClusterIdentifier`), da `docdb-2025-03-27-19-21-15` a`docdb-new-name`.

   ```
   {
       "DBClusterIdentifier": "docdb-new-name"
   }
   ```

2. Annulla la modifica eseguendo il comando CLI modify-db-cluster con`--apply-immediately`, riportando il valore modificato al valore originale. Questo esempio ripristina la modifica del nome in sospeso ripristinando il nome corrente del cluster. `--new-db-cluster-identifier`

   Per Linux, macOS o 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
   ```

   Per 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
   ```

Gestione degli RPO per i cluster globali di Amazon DocumentDB

Con un cluster globale Amazon DocumentDB, puoi gestire l'obiettivo del punto di ripristino (RPO) utilizzando il parametro. `global_db_rpo` RPO rappresenta la quantità massima di dati che possono essere persi in caso di interruzione.

Quando imposti un RPO per il tuo cluster globale Amazon DocumentDB, Amazon DocumentDB monitora il tempo di ritardo dell'RPO di tutti i cluster secondari. Questo monitoraggio garantisce che almeno un cluster secondario rimanga all'interno della finestra RPO di destinazione.

L'impostazione RPO controlla il modo in cui Amazon DocumentDB gestisce le transazioni di scrittura sul cluster primario per limitare la potenziale perdita di dati in caso di failover. Amazon DocumentDB valuta i tempi di ritardo di RPO e RPO per eseguire (o bloccare) le transazioni sul primario nel modo seguente:

- Conferma la transazione se almeno un cluster di database secondario ha un tempo di ritardo RPO inferiore rispetto all'RPO.

- Blocca la transazione se tutti i cluster di database secondari hanno tempi di ritardo RPO superiori all'RPO.

In altre parole, se tutti i cluster secondari sono inferiori all'RPO di destinazione, Amazon DocumentDB sospende le transazioni sul cluster primario. Amazon DocumentDB riprende ed esegue il commit delle transazioni in pausa non appena il ritardo di almeno un cluster DB secondario scende al di sotto dell'RPO. Il risultato è che nessuna transazione può eseguire il commit fino al raggiungimento dell'RPO.

Il parametro `global_db_rpo` è dinamico. Se decidi che non vuoi che tutte le transazioni di scrittura si blocchino fino a quando il ritardo non diminuisce a un livello sufficiente, puoi ripristinarlo rapidamente. In questo caso, Amazon DocumentDB applica la modifica dopo un breve ritardo.

> **Important**
>
> In un database globale con solo due AWS regioni, consigliamo di mantenere il valore predefinito del `global_db_rpo` parametro nel gruppo di parametri della regione secondaria. In caso contrario, l'esecuzione di un failover a causa della perdita della AWS regione primaria potrebbe causare la sospensione delle transazioni da parte di Amazon DocumentDB. Attendi invece che Amazon DocumentDB completi la ricostruzione del cluster nella vecchia AWS regione fallita prima di modificare questo parametro per applicare un RPO massimo.

Argomenti

- Impostazione dell'obiettivo del punto di ripristino
- Visualizzazione dell'obiettivo del punto di ripristino
- Disattivazione dell'obiettivo del punto di ripristino

Impostazione dell'obiettivo del punto di ripristino

Il `global_db_rpo` parametro controlla l'impostazione RPO per un database Amazon DocumentDB. I valori validi vanno da 20 secondi a 2.147.483.647 secondi (68 anni). Scegli un valore realistico per soddisfare le tue esigenze aziendali. Ad esempio, puoi consentire fino a 10 minuti per l'RPO, nel qual caso si imposta il valore su 600.

Puoi impostare questo valore per il tuo cluster globale Amazon DocumentDB utilizzando l'API Console di gestione AWS AWS CLI, la o Amazon DocumentDB.

Using the Console di gestione AWS

Per impostare l'RPO

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Scegli il cluster primario del tuo cluster globale Amazon DocumentDB e apri la scheda Configurazione per trovare il gruppo di parametri del cluster DB.

   I gruppi di parametri non possono essere modificati direttamente. Puoi invece procedere come descritto di seguito:

   - Crea un gruppo di parametri cluster di database personalizzato utilizzando il gruppo di parametri predefinito appropriato come punto di partenza.
   - Nel gruppo di parametri del cluster DB personalizzato, imposta il valore del parametro global_db_rpo in base al tuo caso d'uso. I valori validi vanno da 20 secondi fino al valore intero massimo di 2.147.483.647 (68 anni).
   - Applica il gruppo di parametri del cluster DB modificato al tuo cluster DB Amazon DocumentDB.

Per ulteriori informazioni sulla modifica dei gruppi di parametri del cluster DB, consulta. Modifica dei gruppi di parametri del cluster Amazon DocumentDB

Using the AWS CLI

Per impostare il parametro `global_db_rpo`, utilizza il comando della CLI modify-db-cluster-parameter-group. Nel comando, specificate il nome del gruppo di parametri del cluster primario e i valori per il parametro RPO.

Nell'esempio seguente l'RPO viene impostato su 600 secondi per il gruppo di parametri cluster di database primario denominato `my_custom_global_parameter_group`.

Per Linux, macOS o 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"
```

Per 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

Per modificare il `global_db_rpo` parametro, utilizzate l'operazione ModifyDBClusterParameterGroup API.

Visualizzazione dell'obiettivo del punto di ripristino

L'obiettivo del punto di ripristino (RPO) di un cluster globale è memorizzato nel `global_db_rpo` parametro di ogni cluster DB.

Puoi utilizzare la CLI per visualizzare il `global_db_rpo` parametro per un cluster DB Amazon DocumentDB. Utilizza l'`--query`opzione per restituire solo il `global_db_rpo` parametro dal gruppo di parametri.

Per Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name my_custom_global_parameter_group \
    --query "Parameters[?ParameterName=='global_db_rpo']"
```

Per Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name my_custom_global_parameter_group ^
    --query "Parameters[?ParameterName=='global_db_rpo']"
```

Il comando restituisce un risultato simile al seguente.

```
[
    {
        "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"
    }
]
```

Per ulteriori informazioni sulla visualizzazione dei parametri del gruppo di parametri del cluster, vedereGestione dei gruppi di parametri del cluster Amazon DocumentDB.

Disattivazione dell'obiettivo del punto di ripristino

Per disabilitare l'RPO, reimpostare il parametro `global_db_rpo`. Puoi reimpostare i parametri utilizzando l'API Console di gestione AWS AWS CLI, la o l'API Amazon DocumentDB.

Using the Console di gestione AWS

Per disabilitare l'RPO

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco scegliere il gruppo di parametri cluster DB primario.
4. Scegli il pulsante di opzione accanto al parametro global_db_rpo.
5. Scegliete Ripristina i valori predefiniti e confermatelo.

Per ulteriori informazioni su come reimpostare un parametro con la console, vedereModifica dei gruppi di parametri del cluster Amazon DocumentDB.

Using the AWS CLI

Per reimpostare il parametro `global_db_rpo`, utilizzare il comando reset-db-cluster-parameter-group.

Per Linux, macOS o 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"
```

Per 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

Per reimpostare il `global_db_rpo` parametro, utilizzate l'operazione ResetDBClusterParameterGroup API.

Gestione dei cluster Amazon DocumentDB

Per gestire un cluster Amazon DocumentDB, devi disporre di una policy IAM con le autorizzazioni appropriate del piano di controllo Amazon DocumentDB. Queste autorizzazioni consentono di creare, modificare ed eliminare i cluster e le istanze. La `AmazonDocDBFullAccess` policy fornisce tutte le autorizzazioni necessarie per amministrare un cluster Amazon DocumentDB.

I seguenti argomenti mostrano come eseguire varie attività quando si lavora con i cluster Amazon DocumentDB, tra cui la creazione, l'eliminazione, la modifica, la connessione e la visualizzazione dei cluster.

Argomenti

- Comprendere i cluster
- Impostazioni del cluster Amazon DocumentDB
- Configurazioni di storage del cluster Amazon DocumentDB
- Determinazione dello stato di un cluster
- Ciclo di vita del cluster Amazon DocumentDB
- Scalabilità dei cluster Amazon DocumentDB
- Clonazione di un volume per un cluster Amazon DocumentDB
- Comprendere la tolleranza agli errori del cluster Amazon DocumentDB

Comprendere i cluster

Amazon DocumentDB separa elaborazione e storage e trasferisce la replica e il backup dei dati sul volume del cluster. Un volume cluster fornisce un servizio di storage durevole, affidabile e altamente

disponibile che replica i dati in sei modi in tre zone di disponibilità. Le repliche consentono una maggiore disponibilità dei dati e il dimensionamento della lettura. Ogni cluster può scalare fino a 15 repliche.

Sostantivo	Description	Operazioni API (verbi)

Cluster	È composto da una o più istanze e da un volume di storage cluster per la gestione dei dati di tali istanze.	`create-db-cluster`
		`delete-db-cluster`
		`describe-db-clusters`
		`modify-db-cluster`

Istanza	La lettura e la scrittura dei dati sul volume di storage cluster vengono eseguite mediante le istanze. In un determinato cluster sono disponibili due tipi di istanze: primaria e di replica. Un cluster ha sempre un'istanza primaria e può avere 0-15 repliche.	`create-db-instance`
		`delete-db-instance`
		`describe-db-instances`
		`modify-db-instance`
		`describe-orderable-db-instance-options`
		`reboot-db-instance`

Volume cluster	Un volume di storage per database virtuali che si estende su tre zone di disponibilità, ciascuna delle quali contiene due copie dei dati del cluster.	N/A
Istanza primaria	Supporta operazioni di lettura e scrittura ed esegue tutte le modifiche ai dati del volume cluster. Ogni cluster contiene un'istanza primaria.	N/A

Sostantivo	Description	Operazioni API (verbi)
Istanza di replica	Supporta solo le operazioni di lettura. Ogni cluster Amazon DocumentDB può avere fino a 15 istanze di replica oltre all'istanza primaria. Le repliche distribuiscono il carico di lavoro di lettura. Collocando le repliche in zone di disponibilità separate, puoi anche aumentare la disponibilità del database.	N/A
Endpoint del cluster	Un endpoint per un cluster Amazon DocumentDB che si connette all'istanza primaria corrente del cluster. Ogni cluster Amazon DocumentDB ha un endpoint del cluster e un'istanza primaria.	N/A
Endpoint di lettura	Un endpoint per un cluster Amazon DocumentDB che si connette a una delle repliche disponibili per quel cluster. Ogni cluster Amazon DocumentDB ha un endpoint di lettura. Se è presente più di una replica, l'endpoint del lettore indirizza ogni richiesta di connessione a una delle repliche di Amazon DocumentDB.	N/A

Sostantivo	Description	Operazioni API (verbi)
Endpoint dell'istanza	Un endpoint per un'istanz a in un cluster Amazon DocumentDB che si connette a un'istanza specifica. Ogni istanza di un cluster, a prescindere dal tipo, ha un proprio endpoint dell'istanza esclusivo.	N/A

Impostazioni del cluster Amazon DocumentDB

Quando crei o modifichi un cluster, è importante comprendere quali siano i parametri immutabili e quali quelli modificabili dopo la creazione del cluster. La tabella seguente elenca tutte le impostazioni (parametri) specifiche di un cluster. Come specificato nella tabella, alcuni sono modificabili, altri no.

> **Note**
>
> Queste impostazioni non devono essere confuse con i gruppi di parametri del cluster Amazon DocumentDB e i relativi parametri. Per ulteriori informazioni sui gruppi di parametri del cluster, consulta Gestione dei gruppi di parametri del cluster Amazon DocumentDB.

Parametro	Modificabile	Note
`DBClusterIdentifier`	Sì	Vincoli per la denominazione • La lunghezza è di [1—63] lettere, numeri o trattini. • Il primo carattere deve essere una lettera. • Non può terminare con un trattino o contenere due trattini consecutivi. • Deve essere univoco per tutti i cluster di Amazon RDS, Amazon Neptune e Amazon DocumentDB per regione. Account AWS

Parametro	Modificabile	Note
Engine	No	Deve essere `docdb`.
BackupRetentionPeriod	Sì	Deve essere compreso tra [1 e 35] giorni.
DBClusterParameterGroupName	Sì	Vincoli per la denominazione • La lunghezza è di [1-255] caratteri alfanumerici. • Il primo carattere deve essere una lettera. • Non può terminare con un trattino o contenere due trattini consecutivi.
DBSubnetGroupName	No	Dopo aver creato un cluster, non puoi modificare la sottorete.
EngineVersion	No	Il valore può essere `5.0.0` (impostazione predefinita), o`4.0.0`. `3.6.0`
KmsKeyId	No	Se scegli di crittografare il cluster, non puoi modificare la AWS KMS chiave utilizzata per crittografare il cluster.
MasterUsername	No	Dopo aver creato un cluster, non puoi modificare il `MasterUsername` . Vincoli per la denominazione • La lunghezza è di [1—63] caratteri alfanumerici. • Il primo carattere deve essere una lettera. • Non può essere una parola riservata del motore di database.

Parametro	Modificabile	Note

MasterUserPassword	Sì	Vincoli:
		• La lunghezza è di [8—100] caratteri ASCII stampabili.
		• È possibile utilizzare qualsiasi carattere ASCII stampabile eccetto i seguenti:
		  • **/** (barra)
		  • **"** (virgolette doppie)
		  • **@** (simbolo chiocciola)

Port	Sì	Il numero di porta è valido per tutte le istanze del cluster.
PreferredBackupWindow	Sì	
PreferredMaintenanceWindow	Sì	
StorageEncrypted	No	Se scegli di crittografare il cluster, non potrà essere decrittografato.
StorageType	Sì	Il tipo di archiviazione per il cluster DB: Standard (`standard`) o (). I/O-Optimized `iopt1` Impostazione predefinita: `standard` Questo parametro può essere configurato con `CreateDBCluster eModifyDBCluster` . Per ulteriori informazioni, consulta Configurazioni di storage del cluster Amazon DocumentDB.
Tags	Sì	

Parametro	Modificabile	Note
`VpcSecurityGroupIds`	No	Dopo aver creato un cluster, non puoi modificare il VPC su cui risiede.

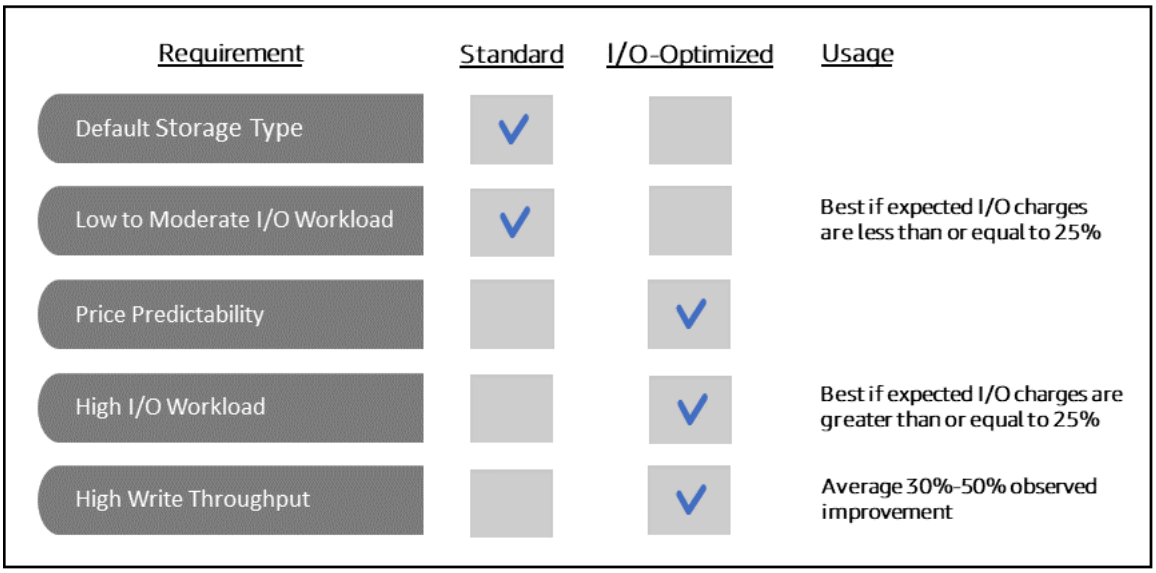
Configurazioni di storage del cluster Amazon DocumentDB

A partire da Amazon DocumentDB 5.0, i cluster basati su istanze supportano due tipi di configurazione di storage:

- Storage standard Amazon DocumentDB: progettato per clienti con consumi da bassi a moderati. I/O Se prevedi che i I/O costi siano inferiori al 25% del totale del cluster Amazon DocumentDB, questa scelta potrebbe essere l'ideale per te. Con la configurazione di storage standard di Amazon DocumentDB, la fatturazione viene calcolata in I/O base al pay-per-request, oltre ai costi di istanza e storage. Ciò significa che la fatturazione potrebbe variare da un ciclo all'altro in base all'utilizzo. La configurazione è personalizzata per soddisfare le I/O esigenze fluttuanti dell'applicazione.
- I/O-optimized Storage Amazon DocumentDB: progettato per i clienti che danno priorità alla prevedibilità dei prezzi o hanno applicazioni intensive. I/O La I/O-optimized configurazione offre prestazioni migliorate, maggiore produttività e latenza ridotta per i clienti con carichi di lavoro intensivi. I/O Se prevedi che i I/O costi superino il 25% dei costi totali del cluster Amazon DocumentDB, questa opzione offre un rapporto prezzo/prestazioni migliorate. Con la configurazione di I/O-optimized storage di Amazon DocumentDB, non ti verrà addebitato alcun costo in base alle I/O operazioni, garantendo costi prevedibili per ogni ciclo di fatturazione. La configurazione stabilizza i costi migliorando al contempo le prestazioni.

Puoi passare i cluster di database esistenti una volta ogni 30 giorni allo storage Amazon I/O-optimized DocumentDB. Puoi tornare allo storage standard di Amazon DocumentDB in qualsiasi momento. La data successiva in cui modificare la configurazione dello storage I/O-optimized può essere tracciata con il `describe-db-clusters` comando utilizzando AWS CLI o tramite la pagina Console di gestione AWS di configurazione del cluster.

Puoi creare un nuovo cluster di database che include la I/O-optimized configurazione di Amazon DocumentDB o convertire i cluster di database esistenti con pochi clic Console di gestione AWS, modificare un singolo parametro in AWS Command Line Interface (AWS CLI) o tramite SDK. AWS Non sono necessari tempi di inattività o riavvio delle istanze durante o dopo la modifica della configurazione di storage.

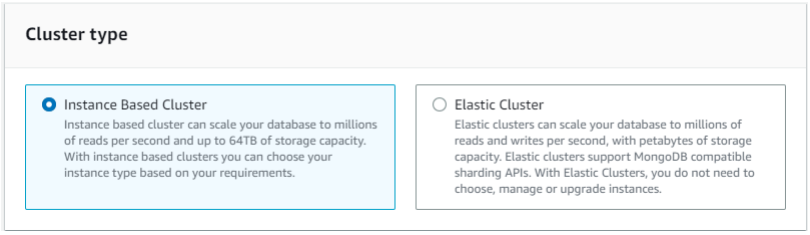


Creazione di un cluster I/O-optimized

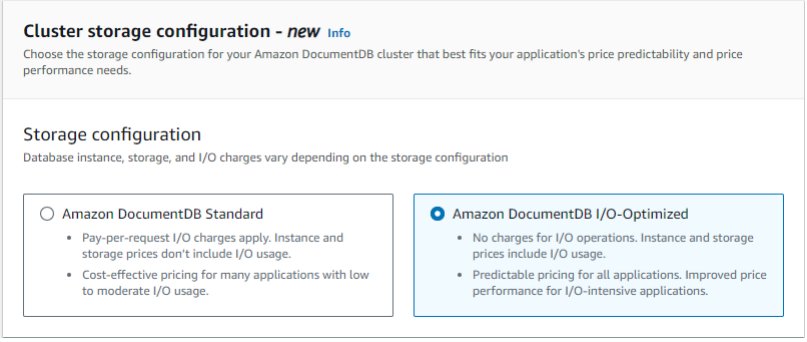
Using the Console di gestione AWS

Per creare o modificare un I/O-optimized cluster utilizzando Console di gestione AWS:

1. Nella console di gestione Amazon DocumentDB, in Clusters, scegli Crea o seleziona il cluster e scegli Azioni, quindi scegli Modifica.
2. Se stai creando un nuovo cluster, assicurati di scegliere Instance Based Cluster nella sezione Tipo di cluster (questa è l'opzione predefinita).



3. Nella sezione Configurazione, in Configurazione dello storage del cluster, scegli Amazon DocumentDB I/O-Optimized.



4. Completa la creazione o la modifica del cluster e scegli Crea cluster o Modifica cluster.

   Per il processo completo di creazione del cluster, consultaCreazione di un cluster e di un'istanza primaria utilizzando Console di gestione AWS.

   Per il processo completo di modifica del cluster, consultaModifica di un cluster Amazon DocumentDB.

Using the AWS CLI

Per creare un I/O-optimized cluster utilizzando AWS CLI:

Negli esempi seguenti, sostituisci ciascuno *user input placeholder* con le tue informazioni.

Per Linux, macOS o 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
```

Per 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
```

Analisi dei costi per determinare la configurazione dello storage

Con Amazon DocumentDB, hai la flessibilità di scegliere la configurazione di storage per ogni cluster di database di cui disponi. Per allocare correttamente i cluster tra standard e I/O-optimized, puoi tenere traccia dei costi di Amazon DocumentDB per cluster. Per fare ciò, puoi aggiungere tag ai cluster esistenti, abilitare i tag di allocazione dei costi nella tua Gestione costi e fatturazione AWS dashboard e analizzare i costi per un determinato cluster nel. AWS Cost Explorer Service Per informazioni sull'analisi dei costi, consulta il nostro blog Utilizzo dei tag di allocazione dei costi.

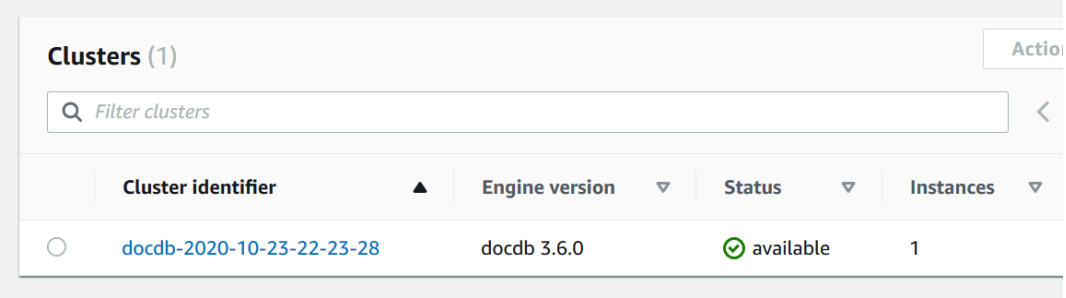
Determinazione dello stato di un cluster

È possibile determinare lo stato di un cluster utilizzando Console di gestione AWS o AWS CLI.

Using the Console di gestione AWS

Utilizza la procedura seguente per visualizzare lo stato del tuo cluster Amazon DocumentDB utilizzando Console di gestione AWS

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.
3. Nella colonna Cluster identifier (Identificatore cluster) individua il nome del cluster desiderato. Quindi, per trovare lo stato del cluster, leggi la riga in corrispondenza della colonna Status (Stato), come illustrato di seguito.



Using the AWS CLI

Usa l'`describe-db-clusters`operazione per visualizzare lo stato del tuo cluster Amazon DocumentDB utilizzando. AWS CLI

Il codice seguente individua lo stato del cluster `sample-cluster`.

Per Linux, macOS o Unix:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster  \
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Per Windows:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster  ^
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
[
    [
        "sample-cluster",
        "available"
    ]
]
```

Ciclo di vita del cluster Amazon DocumentDB

Il ciclo di vita di un cluster Amazon DocumentDB include la creazione, la descrizione, la modifica e l'eliminazione del cluster. Questa sezione fornisce informazioni su come completare questi processi.

Argomenti

- Creazione di un cluster Amazon DocumentDB
- Descrizione dei cluster Amazon DocumentDB
- Modifica di un cluster Amazon DocumentDB
- Determinazione della manutenzione in sospeso

- Esecuzione di un aggiornamento della patch alla versione del motore di un cluster
- Aggiornamento del sistema operativo di un cluster
- Arresto e avvio di un cluster Amazon DocumentDB
- Eliminazione di un cluster Amazon DocumentDB

Creazione di un cluster Amazon DocumentDB

Un cluster Amazon DocumentDB è costituito da istanze e un volume del cluster che rappresenta i dati del cluster. Il volume cluster viene replicato in sei modi su tre zone di disponibilità come un unico volume virtuale. Il cluster contiene un'istanza primaria e, facoltativamente, fino a 15 istanze di replica.

Le seguenti sezioni mostrano come creare un cluster Amazon DocumentDB utilizzando il o il Console di gestione AWS . AWS CLI Puoi quindi aggiungere al cluster istanze di replica aggiuntive. Quando utilizzi la console per creare il tuo cluster Amazon DocumentDB, contemporaneamente viene creata automaticamente un'istanza primaria. Se utilizzi il AWS CLI per creare il tuo cluster Amazon DocumentDB, dopo che lo stato del cluster è disponibile, devi creare l'istanza primaria per quel cluster.

Prerequisiti

Di seguito sono riportati i prerequisiti per la creazione di un cluster Amazon DocumentDB.

Se non ne hai uno Account AWS, completa i seguenti passaggi per crearne uno.

Per iscriverti a Account AWS

1. Aperta https://signin.aws.amazon.com/signup? request_type=registrati.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata o un messaggio di testo e ti verrà chiesto di inserire un codice di verifica attraverso la tastiera del telefono.

   Quando ti iscrivi a un Account AWS, viene creato un. Utente root dell'account AWS L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

Prerequisiti per VPC

Puoi creare un cluster Amazon DocumentDB solo in un Amazon Virtual Private Cloud (Amazon VPC). Il tuo Amazon VPC deve avere almeno una sottorete in ciascuna di almeno due zone di disponibilità per poterlo utilizzare con un cluster Amazon DocumentDB. Distribuendo le istanze del cluster tra le zone di disponibilità, ti assicuri che le istanze siano disponibili nel cluster nell'improbabile caso di errore di una zona di disponibilità.

Prerequisiti della sottorete

Quando crei un cluster Amazon DocumentDB, devi scegliere un VPC e il corrispondente gruppo di sottorete all'interno di quel VPC per avviare il cluster. Le sottoreti determinano la zona di disponibilità e l'intervallo IP all'interno della zona di disponibilità che desideri utilizzare per avviare un'istanza. In questa documentazione i termini sottorete e zona di disponibilità saranno utilizzati indifferentemente. Un gruppo di sottoreti è un set di sottoreti denominato (o zone di disponibilità). Ciò che un gruppo di sottorete ti consente di fare è specificare le zone di disponibilità da utilizzare per avviare le istanze Amazon DocumentDB. Ad esempio, per avere alta disponibilità in un cluster con tre istanze, è consigliabile effettuare il provisioning di ciascuna di queste istanze in zone di disponibilità separate. In questo modo, se una singola zona di disponibilità non funziona, sarà interessata solo una singola istanza.

Le istanze Amazon DocumentDB possono attualmente essere assegnate a un massimo di tre zone di disponibilità. Anche se un gruppo di sottoreti ha più di tre sottoreti, puoi utilizzarne solo tre per creare un cluster Amazon DocumentDB. Di conseguenza, quando crei un gruppo di sottoreti, è consigliabile scegliere solo le tre sottoreti in cui desideri distribuire le tue istanze. Negli Stati Uniti orientali (Virginia settentrionale), il tuo gruppo di sottoreti può avere sei sottoreti (o zone di disponibilità). Tuttavia, quando viene effettuato il provisioning di un cluster Amazon DocumentDB, Amazon DocumentDB sceglie tre di quelle zone di disponibilità che utilizza per il provisioning delle istanze.

Ad esempio, supponiamo che durante la creazione di un cluster, Amazon DocumentDB scelga le zone di disponibilità {1A, 1B e 1C}. Se tenti di creare un'istanza nella zona di disponibilità {1D}, la chiamata API non riesce. Tuttavia, se scegli di creare un'istanza senza specificare una particolare zona di disponibilità, Amazon DocumentDB sceglie una zona di disponibilità per tuo conto. Amazon DocumentDB utilizza un algoritmo per bilanciare il carico delle istanze tra le zone di disponibilità per aiutarti a raggiungere un'elevata disponibilità. Ad esempio, se viene effettuato il provisioning di tre istanze, per impostazione predefinita il provisioning viene effettuato nelle tre zone di disponibilità e non in una sola di esse.

Raccomandazioni:

- A meno che non si abbia un motivo specifico, creare sempre un gruppo di sottoreti con tre sottoreti. In questo modo si assicura che i cluster con tre o più istanze possano ottenere una disponibilità più elevata in quanto ne verrà effettuato il provisioning su tre zone di disponibilità.
- Distribuisci sempre le istanze in più zone di disponibilità per ottenere una disponibilità più elevata. Non posizionare mai tutte le istanze di un cluster in una singola zona di disponibilità.
- Poiché gli eventi di failover possono verificarsi in qualsiasi momento, non devi presumere che un'istanza primaria o le istanze di replica siano sempre in una determinata zona di disponibilità.

Prerequisiti aggiuntivi

Di seguito sono riportati alcuni prerequisiti aggiuntivi per la creazione di un cluster Amazon DocumentDB:

- Se ti connetti AWS utilizzando credenziali AWS Identity and Access Management (IAM), il tuo account IAM deve disporre di politiche IAM che concedano le autorizzazioni necessarie per eseguire le operazioni di Amazon DocumentDB.

  Se utilizzi un account IAM per accedere alla console Amazon DocumentDB, devi prima accedere Console di gestione AWS con il tuo account IAM. Quindi accedi alla console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
- Per personalizzare i parametri di configurazione per il cluster, devi specificare un gruppo di parametri del cluster e un gruppo di parametri con le impostazioni dei parametri richieste. Per informazioni sulla creazione o la modifica di un gruppo di parametri del cluster o un gruppo di parametri, consulta Gestione dei gruppi di parametri del cluster Amazon DocumentDB.
- È necessario determinare il numero di TCP/IP porta che si desidera specificare per il cluster. I firewall di alcune aziende bloccano le connessioni alle porte predefinite per Amazon DocumentDB. Se il firewall della tua azienda blocca la porta predefinita, scegli un'altra porta per il cluster. Tutte le istanze in un cluster utilizzano la stessa porta.

Creazione di un cluster e di un'istanza primaria utilizzando Console di gestione AWS

Le seguenti procedure descrivono come utilizzare la console per avviare un cluster Amazon DocumentDB con una o più istanze.

Crea un cluster: utilizzando le impostazioni predefinite

Per creare un cluster con istanze utilizzando le impostazioni predefinite utilizzando Console di gestione AWS

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Se desideri creare il tuo cluster in una regione Regione AWS diversa da quella degli Stati Uniti orientali (Virginia settentrionale), scegli la regione dall'elenco nella sezione in alto a destra della console.
3. Nel riquadro di navigazione scegliere Clusters (Cluster), quindi Create (Crea).

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

4. Nella pagina Crea cluster Amazon DocumentDB, completa il riquadro di configurazione.

   a. Identificatore del cluster: accetta il nome fornito da Amazon DocumentDB o inserisci un nome per il tuo cluster; ad esempio,. **`sample-cluster`**

      Vincoli per la denominazione del cluster:

      - La lunghezza è di [1—63] lettere, numeri o trattini.
      - Il primo carattere deve essere una lettera.
      - Non può terminare con un trattino o contenere due trattini consecutivi.
      - Deve essere univoco per tutti i cluster di Amazon RDS, Neptune e Amazon DocumentDB per regione. Account AWS

   b. Versione del motore: accetta la versione predefinita del motore 5.0.0 o, facoltativamente, scegli 8.0.0, 4.0.0 o 3.6.0.

   c. Classe di istanza: accettate l'impostazione predefinita o scegliete la classe di istanza `db.r5.large` desiderata dall'elenco.

   d. Numero di istanze: nell'elenco, scegli il numero di istanze che desideri creare con questo cluster. La prima istanza è l'istanza primaria, tutte le altre sono istanze di replica di

sola lettura. È possibile aggiungere ed eliminare le istanze in un secondo momento, se necessario. Per impostazione predefinita, un cluster Amazon DocumentDB viene avviato con tre istanze (una primaria e due repliche).

5. Completa la sezione di configurazione dello storage del cluster.

   Scegli Amazon DocumentDB Standard (impostazione predefinita) o Amazon I/O-Optimized DocumentDB. Per ulteriori informazioni, consulta Configurazioni di storage del cluster Amazon DocumentDB.

6. Compila il riquadro Authentication (Autenticazione).

   a. Nome utente: inserisci un nome per l'utente principale. Per accedere al cluster, è necessario utilizzare il nome utente principale.

      Vincoli principali per la denominazione degli utenti:

      - La lunghezza è di [1—63] caratteri alfanumerici.
      - Il primo carattere deve essere una lettera.
      - Non può essere una parola riservata del motore di database.

   b. Scegliete una delle seguenti opzioni di password:

      - Gestito in AWS Secrets Manager: scegli questa opzione se desideri AWS Secrets Manager gestire automaticamente la password dell'utente principale.

        Se scegli questa opzione, configura la chiave KMS creandone una tua o utilizzando una chiave creata da Secrets Manager.
      - Gestito automaticamente: scegli questa opzione se desideri gestire automaticamente la tua password utente principale. Se scegli questa opzione, inserisci una password per l'utente principale e confermala. Per accedere al cluster, è necessario utilizzare la password per l'utente principale.

        Vincoli per la password:

        - La lunghezza è di [8-100] caratteri ASCII stampabili.
        - È possibile utilizzare qualsiasi carattere ASCII stampabile eccetto i seguenti:
          - **/** (barra)
          - **"** (virgolette doppie)
          - **@** (simbolo chiocciola)

7. Nella parte inferiore della schermata, scegli uno dei seguenti:
   - Per creare subito il cluster, scegli Create cluster (Crea cluster).
   - Per non creare il cluster, scegli Cancel (Annulla).
   - Per configurare ulteriormente il cluster prima di crearlo, scegliere Show additional configurations (Mostra configurazioni aggiuntive), quindi continuare in Crea un cluster: configurazioni aggiuntive.

     Le configurazioni comprese nella sezione Additional Configurations (Configurazioni aggiuntive) sono:

     - Impostazioni di rete: l'impostazione predefinita prevede l'utilizzo del gruppo di sicurezza `default` VPC.
     - Opzioni del cluster: l'impostazione predefinita prevede l'utilizzo della porta 27017 e del gruppo di parametri predefinito.
     - Crittografia: l'impostazione predefinita prevede l'attivazione della crittografia tramite la chiave. `(default) aws/rds`

       > **Important**
       >
       > Una volta che un cluster è crittografato, non può essere decrittografato.

     - Backup: l'impostazione predefinita prevede la conservazione dei backup per 1 giorno e la scelta della finestra di backup da parte di Amazon DocumentDB.
     - Esportazioni di log: l'impostazione predefinita è di non esportare i log di controllo in Logs. CloudWatch
     - Manutenzione: l'impostazione predefinita prevede che Amazon DocumentDB scelga la finestra di manutenzione.
     - Protezione dall'eliminazione: proteggi il tuo cluster dall'eliminazione accidentale. L'impostazione predefinita per i cluster creati utilizzando la console è abilitata.

     Se accetti le impostazioni predefinite ora, potrai modificare la maggior parte di esse successivamente modificando il cluster.
8. Abilita la connessione in entrata per il gruppo di sicurezza del cluster.

   Se non hai modificato le impostazioni predefinite per il tuo cluster, hai creato un cluster utilizzando il gruppo di sicurezza predefinito per il VPC predefinito nella regione specificata. Per

connetterti ad Amazon DocumentDB, devi abilitare le connessioni in entrata sulla porta 27017 (o sulla porta di tua scelta) per il gruppo di sicurezza del cluster.

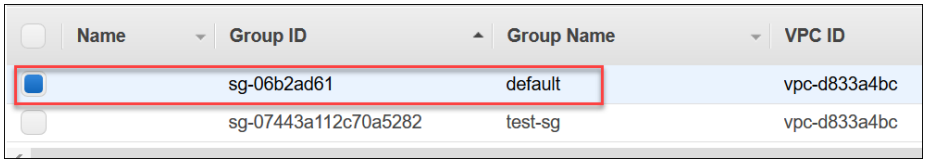
Per aggiungere una connessione in entrata al gruppo di sicurezza del cluster

a. Accedi Console di gestione AWS e apri la console Amazon EC2 all'indirizzo. https://console.aws.amazon.com/ec2/

b. Nella sezione Resources (Risorse) della finestra principale, selezionare Security groups (Gruppi di sicurezza).



c. Dall'elenco di gruppi di sicurezza, individuare il gruppo di sicurezza utilizzato quando si crea il cluster (è probabile che il gruppo di sicurezza predefinito) e scegliere la casella a sinistra del nome del gruppo di sicurezza.



d. Dal menu Actions (Operazioni) , scegliere Edit inbound rules (Modifica regole in entrata), quindi scegliere o inserire i vincoli della regola.

   i. Tipo: dall'elenco, scegli il protocollo da aprire al traffico di rete.

   ii. Protocollo: dall'elenco, scegli il tipo di protocollo.

   iii. Intervallo di porte: per una regola personalizzata, inserisci un numero di porta o un intervallo di porte. Assicurati che il numero di porta o l'intervallo includa la porta specificata quando hai creato il cluster (valore predefinito: 27017).

   iv. Fonte: specifica il traffico che può raggiungere l'istanza. Dall'elenco, scegliere l'origine del traffico. Se scegli Personalizzato, specifica un singolo indirizzo IP o un intervallo di indirizzi IP in notazione CIDR (ad esempio, 203.0.113). 5/32).

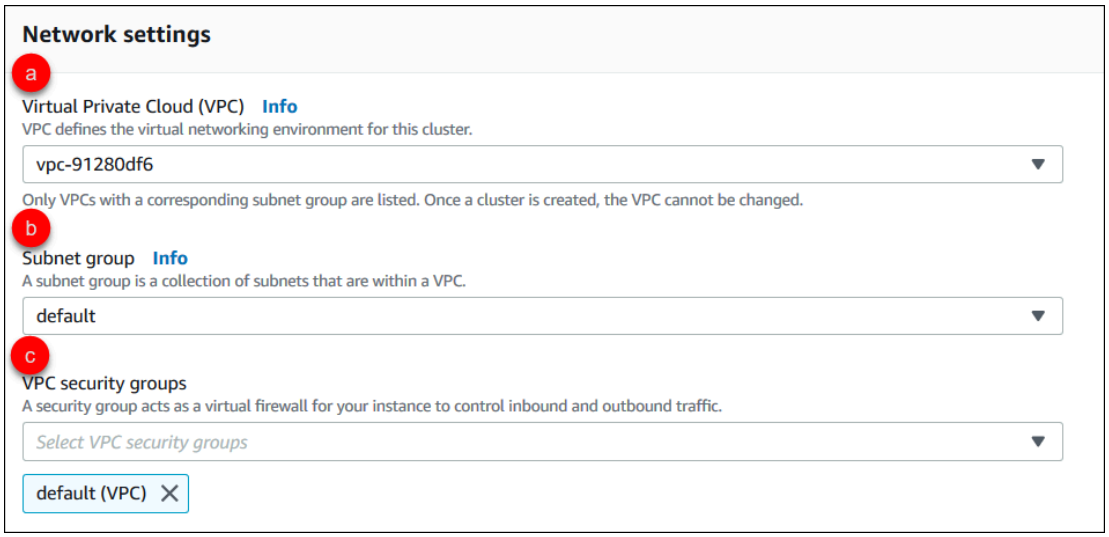
   v. Descrizione: immettere una descrizione per questa regola.

   vi. Dopo aver creato la regola, scegliere Save (Salva).

Crea un cluster: configurazioni aggiuntive

Se desideri accettare le impostazioni predefinite per il cluster, puoi ignorare le seguenti fasi e scegliere Create cluster (Crea cluster).

1. Completa il riquadro Network settings (Impostazioni di rete).

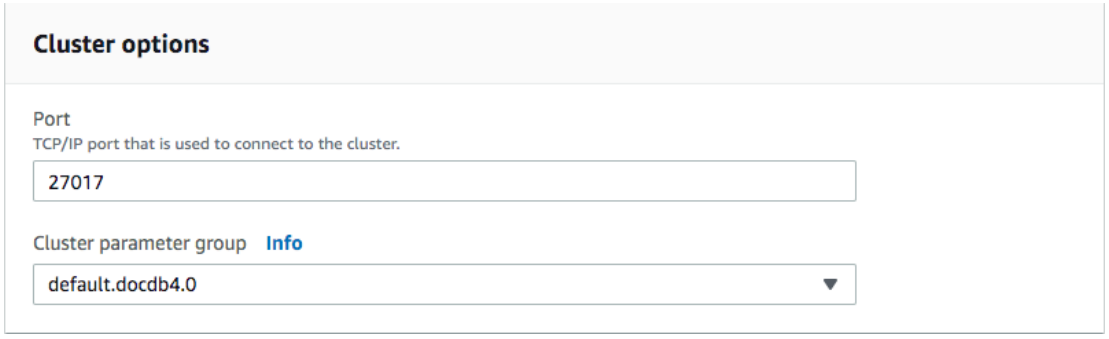


   a. Virtual Private Cloud (VPC): nell'elenco, scegli Amazon VPC in cui desideri avviare questo cluster.

   b. Gruppo di sottorete: nell'elenco, scegli il gruppo di sottorete che desideri utilizzare per questo cluster.

   c. Gruppi di sicurezza VPC: nell'elenco, scegli il gruppo di sicurezza VPC per questo cluster.

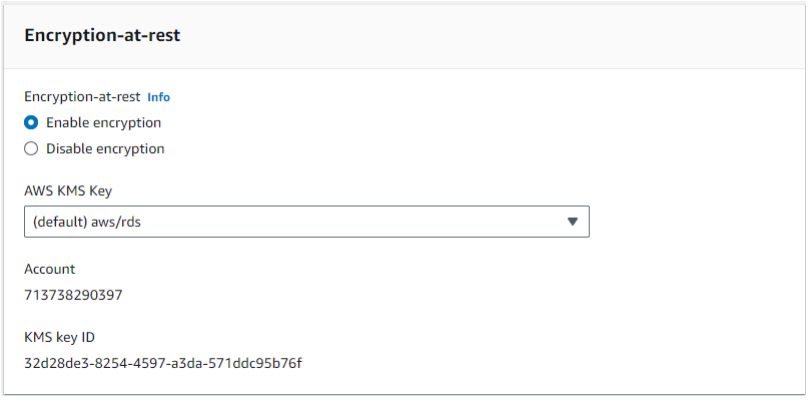
2. Completa il riquadro Cluster options (Opzioni cluster).



   a. Porta del database: utilizza le frecce su e giù per impostare la TCP/IP porta che le applicazioni utilizzeranno per connettersi alla tua istanza.

   b. Gruppo di parametri del cluster: nell'elenco dei gruppi di parametri, scegli il gruppo di parametri del cluster per questo cluster.

3. Completa il riquadro Encryption (Crittografia).



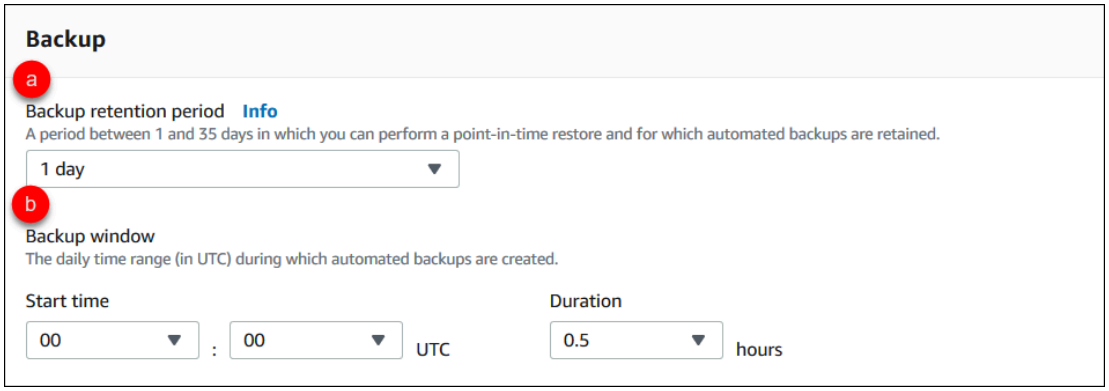
   a. Encryption-at-rest—Scegli una delle seguenti opzioni:

      - Abilita crittografia: impostazione predefinita. Tutti i dati inattivi vengono crittografati. Se scegli di crittografare i dati, non è possibile annullare questa operazione.
      - Disabilita la crittografia: i dati non sono crittografati.

   b. AWS Chiave KMS: disponibile solo se stai crittografando i tuoi dati. Scegliere nell'elenco la chiave che si desidera utilizzare per crittografare i dati nel cluster. Il valore predefinito è `(default) aws/rds`.

      Se si sceglie Enter a key ARN (Immetti ARN chiave), è necessario immettere un Amazon Resource Name (ARN) per la chiave.

4. Completa il riquadro Backup.



   a. Periodo di conservazione dei backup: nell'elenco, scegli il numero di giorni per conservare i backup automatici di questo cluster prima di eliminarli.

   b. Finestra di backup: imposta l'ora e la durata giornaliere durante le quali Amazon DocumentDB deve effettuare i backup di questo cluster.

i. Ora di inizio: nel primo elenco, scegli l'ora di inizio (UTC) per l'avvio dei backup automatici. Dal secondo elenco scegli il minuto dell'ora in cui desideri inizino i backup automatici.

ii. Durata: nell'elenco, scegli il numero di ore da assegnare alla creazione di backup automatici.

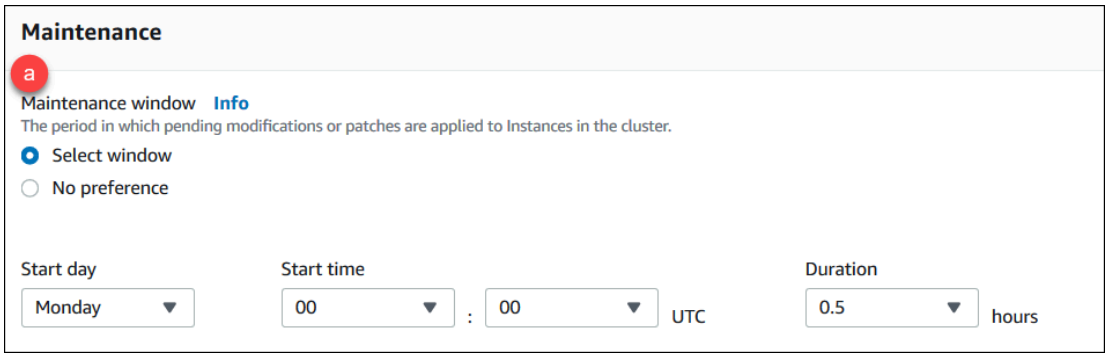
5. Completa il riquadro Esportazioni dei log selezionando i tipi di log che desideri esportare in Logs. CloudWatch



- Log di controllo: seleziona questa opzione per abilitare l'esportazione dei log di controllo in Amazon CloudWatch Logs. Se selezioni Audit logs (Log di audit), devi abilitare `audit_logs` nel gruppo personalizzato di parametri del cluster. Per ulteriori informazioni, consulta Verifica degli eventi di Amazon DocumentDB.
- Log del profiler: seleziona questa opzione per abilitare l'esportazione dei log delle operazioni del profiler in Amazon Logs. CloudWatch Se selezioni Profiler logs (Log del profiler), devi anche modificare i seguenti parametri nel gruppo personalizzato dei parametri del cluster:
  - `profilerenabled`—Impostato su.
  - `profiler_threshold_ms`—Impostato su un valore `[0-INT_MAX]` per impostare la soglia per le operazioni di profilazione.
  - `profiler_sampling_rate`—Impostare un valore per `[0.0-1.0]` impostare la percentuale di operazioni lente da profilare.

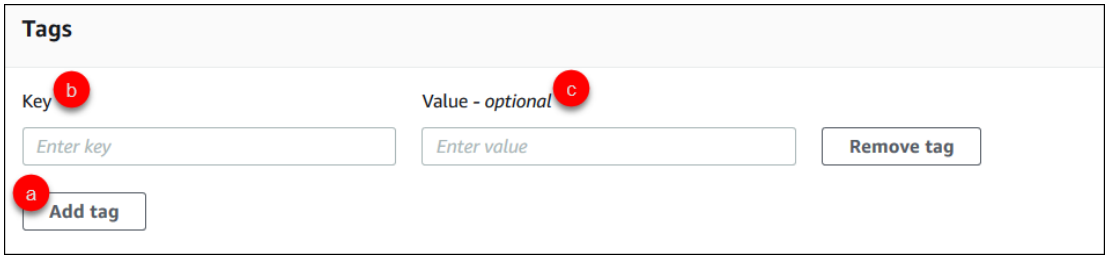
  Per ulteriori informazioni, consulta Profilazione delle operazioni di Amazon DocumentDB.

6. Compilare il riquadro Maintenance (Manutenzione).



- Scegli una delle seguenti opzioni
  - Seleziona finestra: puoi specificare il giorno della settimana, l'ora di inizio UTC e la durata in cui Amazon DocumentDB deve eseguire la manutenzione sul tuo cluster.
    a. Giorno di inizio: nell'elenco, scegli il giorno della settimana in cui iniziare la manutenzione del cluster.
    b. Ora di inizio: negli elenchi, scegli l'ora e il minuto (UTC) per iniziare la manutenzione.
    c. Durata: nell'elenco, scegli quanto tempo dedicare alla manutenzione del cluster. Se la manutenzione non può essere completata nel tempo specificato, il processo di manutenzione proseguirà oltre l'orario indicato fino al completamento.
  - Nessuna preferenza: Amazon DocumentDB sceglie il giorno della settimana, l'ora di inizio e la durata per eseguire la manutenzione.

7. Per aggiungere uno o più tag a questo cluster, completare il riquadro Tags (Tag).

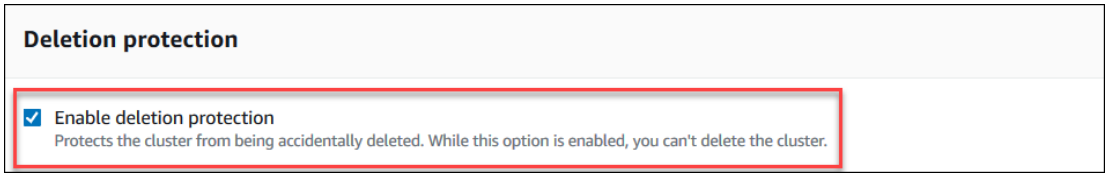


   Per ogni tag che si desidera aggiungere al cluster, ripetere i seguenti passaggi. Puoi averne fino a 10 su un cluster.

   a. Scegli Aggiungi tag.
   b. Digita il tag Key (Chiave).
   c. Opzionalmente digita il Value (Valore) del tag.

   Per rimuovere un tag, scegli Remove (Rimuovi).

8. Quando si crea un cluster utilizzando la console, la Deletion Protection (Protezione dall'eliminazione) viene abilitata per impostazione predefinita. Per disabilitare la protezione dall'eliminazione, deseleziona Enable deletion protection (Abilita la protezione dall'eliminazione). Quando abilitata, la protezione dall'eliminazione impedisce che un cluster venga eliminato. Per eliminare un cluster protetto dall'eliminazione, è necessario prima modificare il cluster per disabilitare la protezione dall'eliminazione.



Per ulteriori informazioni sulla protezione dall'eliminazione, consultare Eliminazione di un cluster Amazon DocumentDB.

9. Per creare il cluster, scegli Create cluster (Crea cluster). Altrimenti, scegli Cancel (Annulla).

Creazione di un cluster utilizzando AWS CLI

Le seguenti procedure descrivono come utilizzare il per AWS CLI avviare un cluster Amazon DocumentDB e creare una replica Amazon DocumentDB.

Parameters

- **`--db-cluster-identifier`**: obbligatorio. Una stringa di lettere minuscole che identifica il cluster.

  Vincoli per la denominazione del cluster:

  - La lunghezza è di [1—63] lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.
  - Deve essere univoco per tutti i cluster (su Amazon RDS, Amazon Neptune e Amazon DocumentDB) per account, per regione. AWS

- **`--engine`**: obbligatorio. Deve essere **`docdb`**.
- **`--deletion-protection | --no-deletion-protection`**—Facoltativo. Quando è abilitata la protezione dall'eliminazione, questa impedisce che il cluster venga eliminato. Quando si utilizza il AWS CLI, l'impostazione predefinita prevede la disattivazione della protezione da eliminazione.

  Per ulteriori informazioni sulla protezione dall'eliminazione, consultare Eliminazione di un cluster Amazon DocumentDB.
- **`--storage-type standard | iopt1`**—Facoltativo. Default: **`standard`**. La configurazione di archiviazione del cluster. I valori validi sono `standard` (Standard) o `iopt1` (I/O-optimized).
- **`--master-username`**: obbligatorio. Il nome utilizzato per autenticare l'utente.

Vincoli per la denominazione dell'utente master:

- La lunghezza è compresa tra 1 e 63 caratteri alfanumerici.
- Il primo carattere deve essere una lettera.
- Non può essere una parola riservata del motore di database.

- **`--master-user-password`**—Facoltativo. La password utilizzata per autenticare l'utente.

  Vincoli per la password master:

  - La lunghezza è di [8-100] caratteri ASCII stampabili.
  - È possibile utilizzare qualsiasi carattere ASCII stampabile eccetto i seguenti:
    - **`/`** (barra)
    - **`"`** (virgolette doppie)
    - **`@`** (simbolo chiocciola)

- **`--manage-master-user-password`**—Opzionale. Amazon DocumentDB genera la password dell'utente principale e la gestisce per tutto il suo ciclo di vita in Secrets Manager.

Per parametri aggiuntivi, consulta CreateDBCluster. https://docs.aws.amazon.com/documentdb/latest/APIReference/API_CreateDBCluster.html

Per avviare un cluster Amazon DocumentDB utilizzando AWS CLI

Per creare un cluster Amazon DocumentDB, chiama. `create-db-cluster` AWS CLI Il AWS CLI comando seguente crea un cluster Amazon DocumentDB denominato `sample-cluster` con la protezione dall'eliminazione abilitata. Per ulteriori informazioni sulla protezione dall'eliminazione, consulta. Eliminazione di un cluster Amazon DocumentDB

Inoltre, `--engine-version` è un parametro opzionale che per impostazione predefinita è l'ultima versione principale del motore. L'attuale versione predefinita del motore è 5.0.0 (nota: Amazon DocumentDB 8.0 è disponibile ma deve essere specificata esplicitamente). Quando vengono rilasciate nuove versioni principali del motore, la versione predefinita del motore `--engine-version` verrà aggiornata per riflettere l'ultima versione principale del motore. Di conseguenza, per i carichi di lavoro di produzione, e in particolare quelli che dipendono da script, automazione

o CloudFormation modelli, specifica esplicitamente la versione `--engine-version` principale prevista.

> **Note**
>
> Se non `vpc-security-group-id` viene specificato un valore `db-subnet-group-name` o, Amazon DocumentDB utilizzerà il gruppo di sottorete e il gruppo di sicurezza Amazon VPC predefiniti per la regione specificata.

Per Linux, macOS o 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
```

Per 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
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "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"
    }
}
```

La creazione del cluster richiede diversi minuti. Puoi utilizzare Console di gestione AWS o AWS CLI per monitorare lo stato del tuo cluster. Per ulteriori informazioni, consulta Monitoraggio dello stato di un cluster Amazon DocumentDB.

> **Important**
>
> Quando si utilizza il AWS CLI per creare un cluster Amazon DocumentDB, non viene creata alcuna istanza. Di conseguenza, devi creare esplicitamente un'istanza primaria e le eventuali istanze di replica di cui hai bisogno. Puoi utilizzare la console o AWS CLI creare le istanze. Per ulteriori informazioni, consulta Aggiungere un'istanza Amazon DocumentDB a un cluster.

Per ulteriori informazioni, consulta CreateDBCluster nell'API Reference di Amazon DocumentDB.

Descrizione dei cluster Amazon DocumentDB

Puoi utilizzare la Console di gestione Amazon DocumentDB o la AWS CLI per visualizzare dettagli come endpoint di connessione, gruppi di sicurezza, VPC e gruppi di parametri relativi ai tuoi cluster Amazon DocumentDB.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Monitoraggio dello stato di un cluster Amazon DocumentDB
- Individuazione degli endpoint di un cluster

Using the Console di gestione AWS

Utilizza la seguente procedura per visualizzare i dettagli di un cluster Amazon DocumentDB specificato utilizzando la console.

1. Accedi e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nell'elenco dei cluster scegli il nome del cluster di cui desideri visualizzare i dettagli. Le informazioni sul cluster vengono organizzate nei seguenti gruppi:

   - Riepilogo: informazioni generali sul cluster, tra cui la versione del motore, lo stato del cluster, la manutenzione in sospeso e lo stato del relativo gruppo di parametri.
   - Connettività e sicurezza: la sezione Connect elenca gli endpoint di connessione per connettersi a questo cluster con la shell mongo o con un'applicazione. Nella sezione Security Groups (Gruppi di sicurezza) sono elencati i gruppi di protezione associati a questo cluster e l'ID VPC e le descrizioni.

- Configurazione: la sezione Dettagli del cluster elenca i dettagli sul cluster, inclusi Amazon Resource Name (ARN), l'endpoint e il gruppo di parametri del cluster. Vengono inoltre elencate le informazioni di backup del cluster, i dettagli di manutenzione e le impostazioni di sicurezza e di rete. La sezione Cluster instances (Istanze del cluster) elenca le istanze che appartengono a questo cluster con lo stato del ruolo e del gruppo di parametri del cluster di ogni istanza.
- Monitoraggio: le metriche di Amazon CloudWatch Logs per questo cluster. Per ulteriori informazioni, consulta Monitoraggio di Amazon DocumentDB con CloudWatch.
- Eventi e tag: la sezione Eventi recenti elenca gli eventi recenti per questo cluster. Amazon DocumentDB registra gli eventi relativi ai cluster, alle istanze, alle istantanee, ai gruppi di sicurezza e ai gruppi di parametri del cluster. Queste informazioni includono la data, l'ora e il messaggio associati a ciascun evento. La sezione Tag elenca i tag allegati al cluster.

Using the AWS CLI

Per visualizzare i dettagli dei tuoi cluster Amazon DocumentDB utilizzando il AWS CLI`describe-db-clusters` comando come mostrato negli esempi seguenti. Per ulteriori informazioni, consulta DescribedBClusters nel riferimento alle API di gestione delle risorse di Amazon DocumentDB.

> **Note**
>
> Per alcune funzionalità di gestione, come la gestione del ciclo di vita di cluster e istanze, Amazon DocumentDB sfrutta la tecnologia operativa condivisa con Amazon RDS. Il parametro `filterName=engine,Values=docdb` filter restituisce solo i cluster Amazon DocumentDB.

Example

Esempio 1: elenca tutti i cluster Amazon DocumentDB

Il AWS CLI codice seguente elenca i dettagli di tutti i cluster Amazon DocumentDB in una regione.

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
```

```
    "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

Esempio 2: elenca tutti i dettagli per un cluster Amazon DocumentDB specificato

Il AWS CLI codice seguente elenca i dettagli del cluster`sample-cluster`.

Per Linux, macOS o Unix:

```
aws docdb describe-db-clusters \
   --filter Name=engine,Values=docdb \
   --db-cluster-identifier sample-cluster
```

Per Windows:

```
aws docdb describe-db-clusters ^
   --filter Name=engine,Values=docdb ^
   --db-cluster-identifier sample-cluster
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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

Esempio 3: Elenca i dettagli specifici per un cluster Amazon DocumentDB

Per elencare un sottoinsieme dei dettagli dei cluster utilizzando il AWS CLI, aggiungi uno `--query` che specifichi quali membri del cluster deve elencare l'`describe-db-clusters`operazione. Il parametro `--db-cluster-identifier` è l'identificatore per il particolare cluster del quale desideri visualizzare i dettagli. Per ulteriori informazioni sulle query, consulta Come filtrare l'output con l'`--query`opzione nella Guida per l'utente. AWS Command Line Interface

L'esempio seguente elenca le istanze in un cluster Amazon DocumentDB.

Per Linux, macOS o Unix:

```
aws docdb describe-db-clusters \
    --filter Name=engine,Values=docdb \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterMembers]'
```

Per Windows:

```
aws docdb describe-db-clusters ^
    --filter Name=engine,Values=docdb ^
    --db-cluster-identifier sample-cluster ^
    --query 'DBClusters[*].[DBClusterMembers]'
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
[
    [
        [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
```

```
            },
            {
                "DBInstanceIdentifier": "sample-instance-2",
                "IsClusterWriter": false,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            }
        ]
    ]
]
```

Modifica di un cluster Amazon DocumentDB

Per modificare un cluster, il cluster deve essere nello stato disponibile. Non è possibile modificare un cluster arrestato. Se il cluster viene arrestato, per prima cosa avviare il cluster, attendere che diventi disponibile, quindi apportare le modifiche desiderate. Per ulteriori informazioni, consulta Arresto e avvio di un cluster Amazon DocumentDB.

Using the Console di gestione AWS

Utilizza la seguente procedura per modificare un cluster Amazon DocumentDB specifico utilizzando la console.

Come modificare un cluster Amazon DocumentDB

1. Accedi e apri la Console di gestione AWS console di Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Specifica il cluster da modificare scegliendo il pulsante a sinistra del nome del cluster.
4. Scegli Actions (Operazioni), quindi Modify (Modifica).

5. Nel riquadro Modify Cluster: cluster-name (Modifica cluster: cluster-name) apporta le modifiche desiderate. È possibile modificare le seguenti aree:

   - Specifiche del cluster: nome del cluster, gruppi di sicurezza e gestione delle credenziali.
   - Configurazione dello storage del cluster: la modalità di archiviazione dei dati del cluster. Scegli tra Standard e I/O-Optimized configurazione.
   - Opzioni del cluster: la porta e il gruppo di parametri del cluster.
   - Backup: periodo di conservazione del backup e finestra di backup del cluster.
   - Esportazioni di log: abilita o disabilita l'esportazione dei log di controllo o di profilazione.
   - Manutenzione: imposta la finestra di manutenzione del cluster.
   - Protezione da eliminazione: abilita o disabilita la protezione dall'eliminazione sul cluster. Per impostazione predefinita, la protezione da eliminazione è abilitata.
6. Al termine, scegli Continue (Continua) per visualizzare un riepilogo delle modifiche.
7. Se accetti le modifiche, puoi scegliere Modify cluster (Modifica cluster) per modificare il cluster. In alternativa, puoi scegliere Back (Indietro) o Cancel (Annulla) per modificare o annullare le modifiche, rispettivamente.

L'applicazione delle modifiche richiede qualche minuto. Puoi utilizzare il cluster solo quando ha lo stato disponibile. Puoi monitorare lo stato del cluster con la console o l' AWS CLI. Per ulteriori informazioni, consulta Monitoraggio dello stato di un cluster Amazon DocumentDB.

Using the AWS CLI

Utilizzare l'operazione `modify-db-cluster` per modificare il cluster specificato utilizzando il comando AWS CLI. Per ulteriori informazioni, consulta ModifyDBCluster nell'API Reference di Amazon DocumentDB.

Parameters

- **`--db-cluster-identifier`**: obbligatorio. L'identificatore del cluster Amazon DocumentDB che intendi modificare.
- **`--backup-retention-period`**—Facoltativo. Il numero di giorni durante i quali vengono conservati i backup automatici. I valori validi sono 1—35.
- **`--storage-type`**—Facoltativo. La configurazione di archiviazione del cluster. I valori validi sono `standard` (Standard) o `iopt1` (I/O-optimized).

- **`--db-cluster-parameter-group-name`**—Facoltativo. Il nome del gruppo di parametri del cluster da utilizzare per il cluster.
- **`--manage-master-user-password`**—Opzionale. Amazon DocumentDB genera la password dell'utente principale e la gestisce per tutto il suo ciclo di vita in Secrets Manager.
- **`--rotate-master-user-password`**—Facoltativo. Secrets Manager genera una nuova versione segreta per il segreto esistente. La nuova versione del segreto contiene la nuova password dell'utente principale. Amazon DocumentDB modifica la password dell'utente principale del cluster in modo che corrisponda alla password della nuova versione segreta.

  È necessario specificare l'**`--apply-immediately`**opzione quando si ruota la password principale.
- **`--master-user-password`**—Facoltativo. La nuova password per l'utente principale del database.

  Vincoli per la password:

  - La lunghezza è di [8—100] caratteri ASCII stampabili.
  - È possibile utilizzare qualsiasi carattere ASCII stampabile eccetto i seguenti:
    - **`/`** (barra)
    - **`"`** (virgolette doppie)
    - **`@`** (simbolo chiocciola)
- **`--new-db-cluster-identifier`**—Facoltativo. Il nuovo identificatore per il cluster quando un cluster viene rinominato. Questo valore è archiviato come stringa in caratteri minuscoli.

  Vincoli per la denominazione

  - La lunghezza è di [1—63] lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.
  - Deve essere univoco per tutti i cluster di Amazon RDS, Amazon Neptune e Amazon DocumentDB per regione. Account AWS
- **`--preferred-backup-window`**—Facoltativo. L'intervallo temporale giornaliero durante il quale vengono creati i backup automatici, in UTC.
  - Formato: `hh24:mm-hh24:mm`
- **`--preferred-maintenance-window`**—Opzionale. L'intervallo temporale settimanale durante il quale può avvenire la manutenzione dei sistemi, in UTC.

  - Formato: `ddd:hh24:mm-ddd:hh24:mm`
  - Giorni validi: `Sun`, `Mon`, `Tue`, `Wed`, `Thu`, `Fri` e `Sat`.
- **`--deletion-protection`**o **`--no-deletion-protection`** —Facoltativo. Se la protezione dall'eliminazione deve essere abilitata su questo cluster. La protezione dall'eliminazione impedisce che un cluster venga accidentalmente eliminato finché il cluster non viene modificato per disabilitare la protezione dall'eliminazione. Per ulteriori informazioni, consulta Eliminazione di un cluster Amazon DocumentDB.
- **`--apply-immediately`**o**`--no-apply-immediately`**: utilizzare `--apply-immediately` per apportare immediatamente la modifica. Utilizzare `--no-apply-immediately` per apportare la modifica durante la finestra di manutenzione successiva del cluster.

Example

Il codice seguente cambia il periodo di retention dei backup per il cluster `sample-cluster`.

Per Linux, macOS o Unix:

```
aws docdb modify-db-cluster \
       --db-cluster-identifier sample-cluster \
       --apply-immediately \
       --backup-retention-period 7
```

Per Windows:

```
aws docdb modify-db-cluster ^
       --db-cluster-identifier sample-cluster ^
       --apply-immediately ^
       --backup-retention-period 7
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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
    }
}
```

L'applicazione delle modifiche richiede qualche minuto. Puoi utilizzare il cluster solo quando ha lo stato disponibile. Puoi monitorare lo stato del cluster con la console o l' AWS CLI. Per ulteriori informazioni, consulta Monitoraggio dello stato di un cluster Amazon DocumentDB.

Determinazione della manutenzione in sospeso

La manutenzione comporta spesso l'aggiornamento delle seguenti risorse del cluster:

- Sistema operativo (OS) delle istanze nel cluster
- Versione del motore del cluster

Per ulteriori informazioni sulla manutenzione di Amazon DocumentDB, consultaManutenzione di Amazon DocumentDB.

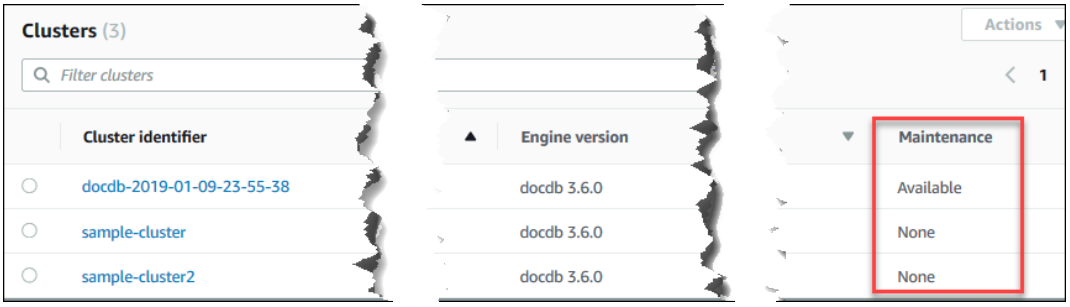
Using the Console di gestione AWS

Puoi utilizzare il Console di gestione AWS per determinare se un cluster ha una manutenzione in sospeso.

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.

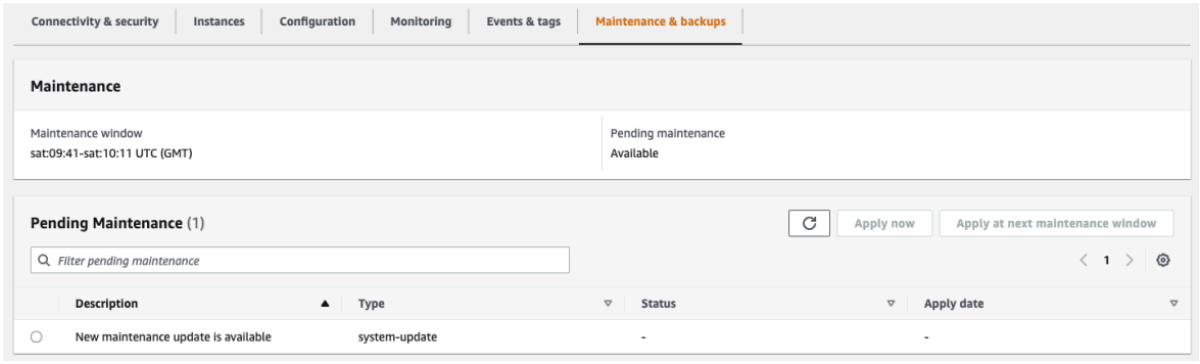
   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Individuare la colonna Maintenance (Manutenzione) per determinare se un cluster è in attesa di manutenzione.



Nessuno indica che il cluster sta eseguendo l'ultima versione della patch del motore e che le sue istanze dispongono degli ultimi aggiornamenti del sistema operativo. Disponibile indica che il cluster è in attesa di manutenzione, il che potrebbe significare che è necessario un aggiornamento del motore o del sistema operativo.

4. Se il cluster ha una manutenzione in sospeso, puoi visualizzare le azioni di manutenzione in sospeso scegliendo il nome del cluster e selezionando la scheda Manutenzione e backup. La sezione Manutenzione in sospeso elenca le azioni di manutenzione in sospeso.



Using the AWS CLI

Puoi utilizzare il AWS CLI per determinare se un cluster dispone della versione del motore e degli aggiornamenti del sistema operativo più recenti utilizzando l'`describe-pending-maintenance-actions`operazione con i seguenti parametri.

Parameters

- **`--resource-identifier`**—Facoltativo. L'ARN per la risorsa (cluster). Se questo parametro viene omesso, vengono elencate le operazioni di manutenzione in attesa per tutti i cluster.
- **`--region`**—Opzionale. L'endpoint della regione AWS in cui eseguire l'operazione, ad esempio `us-east-1`.

Example

Per Linux, macOS o Unix:

```
aws docdb describe-pending-maintenance-actions \
   --resource-identifier arn:aws:rds:aa-example-1:111122223333:cluster:sample-
cluster \
   --region aa-example-1
```

Per Windows:

```
aws docdb describe-pending-maintenance-actions ^
   --resource-identifier arn:aws:rds:aa-example-1:111122223333:cluster:sample-
cluster ^
   --region aa-example-1
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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"
                }
            ]
        }
    ]
}
```

I seguenti tipi di azioni di manutenzione si applicano ai cluster Amazon DocumentDB:

- `system-update`— Aggiorna la patch del motore per il cluster Amazon DocumentDB.

  Se sul tuo cluster è disponibile un aggiornamento della patch del motore in sospeso, continua con i passaggi indicati. Esecuzione di un aggiornamento della patch alla versione del motore di un cluster

- `os-upgrade`— Aggiorna i sistemi operativi di tutte le istanze DB nel cluster Amazon DocumentDB, utilizzando aggiornamenti continui.

  Per ulteriori informazioni, consulta Aggiornamenti del sistema operativo Amazon DocumentDB.

Esecuzione di un aggiornamento della patch alla versione del motore di un cluster

In questa sezione, spiegheremo come distribuire un aggiornamento della patch utilizzando Console di gestione AWS o. AWS CLI Un aggiornamento di patch è un aggiornamento all'interno della stessa versione del motore (ad esempio, l'aggiornamento di una versione del motore 3.6 a una versione del motore 3.6 più recente). Puoi aggiornarlo immediatamente o durante la prossima finestra di manutenzione del cluster. Per determinare se il tuo motore necessita di un aggiornamento, consultaDeterminazione della manutenzione in sospeso. Tieni presente che quando applichi l'aggiornamento, il cluster subirà alcuni tempi di inattività.

> **Note**
>
> Se stai cercando di eseguire l'aggiornamento da una versione principale del motore a un'altra, ad esempio da 3.6 a 5.0 o da 5.0 a 8.0, vedi uno o. Aggiornamento della versione principale in-place di Amazon DocumentDB Aggiornamento del cluster Amazon DocumentDB tramite AWS Database Migration Service

Esistono due requisiti di configurazione per ottenere gli ultimi aggiornamenti delle patch per la versione del motore di un cluster:

- Lo stato del cluster deve essere disponibile.
- Il cluster deve eseguire una versione precedente del motore.

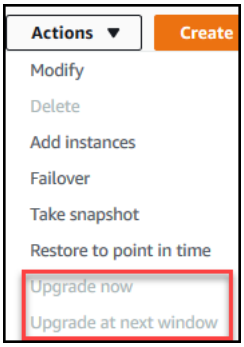
Using the Console di gestione AWS

La procedura seguente applica gli aggiornamenti delle patch alla versione del motore del cluster utilizzando la console. È possibile effettuare l'aggiornamento immediatamente o durante la successiva finestra di manutenzione del cluster.

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.
2. Nel pannello di navigazione scegliere Cluster. Dall'elenco di cluster scegliere il pulsante a sinistra del cluster che si desidera aggiornare. Lo stato del cluster deve essere disponibile.

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nel menu Actions (Operazioni) scegliere una delle opzioni indicate di seguito. Queste opzioni di menu sono selezionabili solo se sul cluster scelto non è in esecuzione la versione più recente del motore.



   - Esegui l'upgrade ora: avvia immediatamente il processo di aggiornamento. Il cluster sarà temporaneamente offline durante l'aggiornamento alla versione più recente del motore.
   - Aggiorna nella finestra successiva: avvia il processo di aggiornamento durante la successiva finestra di manutenzione del cluster. Il cluster sarà temporaneamente offline durante l'aggiornamento del cluster alla versione più recente del motore.
4. Quando si apre la finestra di conferma, scegliere una delle seguenti opzioni:
   - Aggiornamento: per aggiornare il cluster alla versione più recente del motore in base alla pianificazione scelta nel passaggio precedente.

- Annulla: per annullare l'aggiornamento del motore del cluster e continuare con la versione corrente del motore del cluster.

Using the AWS CLI

È possibile applicare gli aggiornamenti delle patch al cluster utilizzando l'`apply-pending-maintenance-action`operazione AWS CLI e con i seguenti parametri.

Parameters

- **`--resource-identifier`**: obbligatorio. L'ARN del cluster Amazon DocumentDB che intendi aggiornare.
- **`--apply-action`**: obbligatorio. `system-update`Da utilizzare per applicare un aggiornamento della patch del motore al cluster.
- **`--opt-in-type`**: obbligatorio. Sono consentiti i seguenti valori.
  - `immediate`—Applica immediatamente l'azione di manutenzione.
  - `next-maintenance`—Applica l'azione di manutenzione durante la finestra di manutenzione successiva.
  - `undo-opt-in`—Annulla qualsiasi richiesta di `next-maintenance` opt-in esistente.

Example

L'esempio seguente applica l'aggiornamento della patch del motore a. `sample-cluster`

Per Linux, macOS o 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
```

Per 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
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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"
            }
        ]
    }
}
```

Aggiornamento del sistema operativo di un cluster

È possibile applicare gli aggiornamenti del sistema operativo (OS) a tutte le istanze del cluster utilizzando l'azione di manutenzione a livello di cluster`os-upgrade`. Amazon DocumentDB aggiorna le istanze in modo continuativo, alcune alla volta, e aggiorna l'istanza principale per ultima per ridurre al minimo i failover. L'aggiornamento viene eseguito durante la finestra di manutenzione del cluster. Dopo che un'istanza riceve un aggiornamento del sistema operativo, la cache del buffer si svuota e le query su tale istanza possono subire una latenza maggiore fino a quando il working set non viene ripopolato dal volume di archiviazione.

Per determinare se il cluster ha un aggiornamento del sistema operativo in sospeso, consulta. Determinazione della manutenzione in sospeso Per i dettagli completi sugli aggiornamenti del sistema operativo, incluse le procedure relative alla console e all'interfaccia della riga di comando, consulta. Aggiornamenti del sistema operativo Amazon DocumentDB

Arresto e avvio di un cluster Amazon DocumentDB

L'arresto e l'avvio dei cluster Amazon DocumentDB possono aiutarti a gestire i costi per gli ambienti di sviluppo e test. Invece di creare ed eliminare cluster e istanze ogni volta che usi Amazon DocumentDB, puoi interrompere temporaneamente tutte le istanze del cluster quando non sono necessarie. È quindi possibile avviarle nuovamente quando è necessario riprendere il testing.

Argomenti

- Panoramica dell'avvio e dell'arresto di un cluster.
- Operazioni che è possibile eseguire su un cluster interrotto

Panoramica dell'avvio e dell'arresto di un cluster.

Nei periodi in cui non è necessario un cluster Amazon DocumentDB, puoi interrompere tutte le istanze di quel cluster contemporaneamente. È possibile avviare nuovamente il cluster ogni volta che è necessario utilizzarlo. L'avvio e l'arresto semplificano i processi di impostazione e rimozione dei cluster utilizzati per lo sviluppo, i test o attività analoghe che non richiedono una disponibilità continua. Puoi interrompere e avviare un cluster utilizzando Console di gestione AWS o AWS CLI con una sola azione, indipendentemente dal numero di istanze presenti nel cluster.

Quando il cluster viene arrestato, il volume di archiviazione associato al cluster rimane invariato. Vengono addebitati solo i costi per lo storage, gli snapshot manuali e lo storage di backup automatici all'interno della finestra di retention specificata. Non ti viene addebitato alcun costo per le ore di istanza. Amazon DocumentDB avvia automaticamente il cluster dopo sette giorni in modo che non subisca gli aggiornamenti di manutenzione richiesti. Quando il cluster viene avviato dopo 7 giorni, avrà nuovamente inizio l'addebito relativo alle istanze del cluster. Mentre il cluster viene arrestato, non è possibile eseguire query sul volume di archiviazione perché l'esecuzione di query richiede che le istanze siano nello stato disponibile.

Quando un cluster Amazon DocumentDB viene interrotto, né il cluster né le relative istanze possono essere modificati in alcun modo. Ciò include l'aggiunta o la rimozione di istanze o l'eliminazione del cluster.

Using the Console di gestione AWS

La seguente procedura mostra come arrestare un cluster con una o più istanze in stato disponibile o avviare un cluster arrestato.

Per arrestare o avviare un cluster Amazon DocumentDB

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.

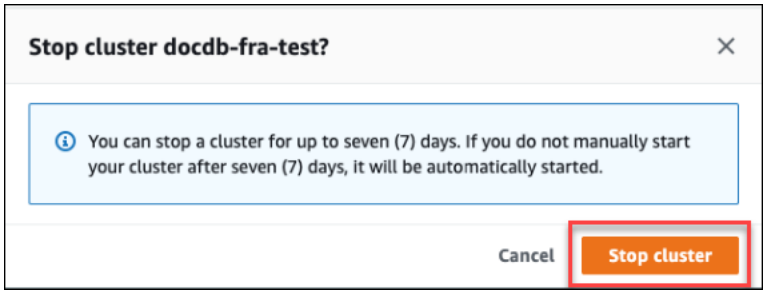
> **Tip**
>
> Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Quindi, nell'elenco dei cluster, scegliere il pulsante a sinistra del nome del cluster che si desidera arrestare o avviare.
4. Scegliere Actions (Operazioni) e quindi scegliere l'operazione che si desidera eseguire sul cluster.
   - Se si desidera arrestare il cluster e il cluster è disponibile:
     a. Scegli Stop (Arresta).

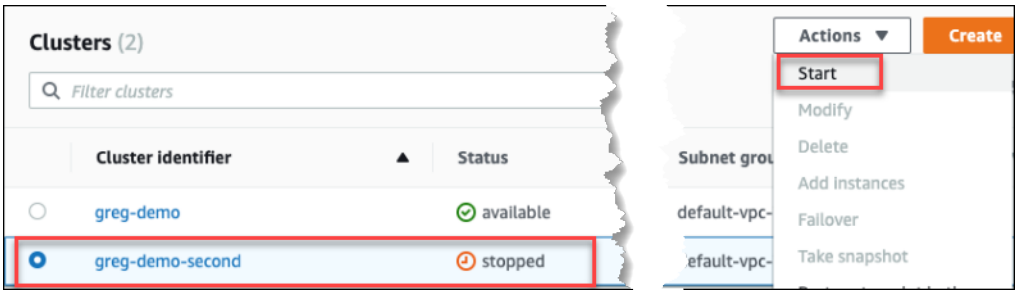


Per evitare l'attivazione del meccanismo di failover, l'operazione di arresto interrompe prima le istanze di replica e poi l'istanza primaria.

     b. Nella finestra di dialogo di conferma, confermare che si desidera arrestare il cluster scegliendo Stop cluster (Arresta cluster) oppure, per mantenere il cluster in esecuzione, scegliere Cancel (Annulla).



- Se si desidera avviare il cluster e il cluster è arrestato, scegliere Start (Avvia).



5. Monitorare lo stato del cluster e delle relative istanze. Se si avvia il cluster, è possibile riprendere ad utilizzare il cluster quando il cluster e le relative istanze risultano disponibili. Per ulteriori informazioni, consulta Determinazione dello stato di un cluster.



Using the AWS CLI

I seguenti codici di esempio illustrano come arrestare un cluster con una o più istanze in stato disponibile o avviare un cluster arrestato.

Per interrompere un cluster con una o più istanze disponibili utilizzando il AWS CLI, usa l'`stop-db-cluster`operazione. Per avviare un cluster arrestato, utilizzare l'operazione `start-db-cluster`. Entrambe le operazioni consentono di utilizzare il parametro `--db-cluster-identifier`.

Parametro:

- **`--db-cluster-identifier`**: obbligatorio. Il nome del cluster da arrestare o avviare.

Example— Per arrestare un cluster utilizzando AWS CLI

Il codice seguente arresta il cluster `sample-cluster`. Il cluster deve avere una o più istanze in stato disponibile.

Per Linux, macOS o Unix:

```
aws docdb stop-db-cluster \
   --db-cluster-identifier sample-cluster
```

Per Windows:

```
aws docdb stop-db-cluster ^
   --db-cluster-identifier sample-cluster
```

Example— Per avviare un cluster utilizzando AWS CLI

Il codice seguente avvia il cluster `sample-cluster`. Il cluster deve attualmente essere arrestato.

Per Linux, macOS o Unix:

```
aws docdb start-db-cluster \
   --db-cluster-identifier sample-cluster
```

Per Windows:

```
aws docdb start-db-cluster ^
   --db-cluster-identifier sample-cluster
```

Operazioni che è possibile eseguire su un cluster interrotto

Quando un cluster Amazon DocumentDB è interrotto, puoi eseguire un ripristino point-in-time in qualsiasi momento all'interno della finestra di conservazione automatica dei backup specificata. Per i dettagli su come eseguire un ripristino point-in-time, consulta Ripristino a un determinato punto nel tempo.

Non puoi modificare la configurazione di un cluster Amazon DocumentDB o di una delle sue istanze mentre il cluster è fermo. Inoltre, non è possibile aggiungere o rimuovere istanze di database dal

cluster o eliminarlo se possiede ancora istanze di database associate. Devi avviare il cluster prima di eseguire questo tipo di operazioni amministrative.

Amazon DocumentDB applica qualsiasi manutenzione programmata al cluster interrotto solo dopo il riavvio. Dopo sette giorni, Amazon DocumentDB avvia automaticamente un cluster interrotto in modo che non rimanga troppo indietro nel suo stato di manutenzione. Quando il cluster viene riavviato, avrà nuovamente inizio l'addebito relativo alle istanze del cluster.

Mentre un cluster è fermo, Amazon DocumentDB non esegue alcun backup automatico né estende il periodo di conservazione del backup.

Eliminazione di un cluster Amazon DocumentDB

Puoi eliminare un cluster Amazon DocumentDB utilizzando Console di gestione AWS o. AWS CLI Per eliminare un cluster, il cluster deve trovarsi nello stato disponibile e non deve avere alcuna istanza associata. Se il cluster viene arrestato, per prima cosa avviare il cluster, attendere che diventi disponibile, quindi eliminarlo. Per ulteriori informazioni, consulta Arresto e avvio di un cluster Amazon DocumentDB.

Protezione da eliminazione

Per proteggere il cluster dall'eliminazione accidentale, è possibile abilitare la protezione dall'eliminazione . Quando si crea un cluster utilizzando la console, la protezione dall'eliminazione viene abilitata per impostazione predefinita. Tuttavia, la protezione dall'eliminazione è disabilitata per impostazione predefinita se si crea un cluster utilizzando la AWS CLI.

Amazon DocumentDB applica la protezione dall'eliminazione per un cluster indipendentemente dal fatto che l'operazione di eliminazione venga eseguita utilizzando la console o il. AWS CLI Se la protezione dall'eliminazione è abilitata, non puoi eliminare un cluster. Per eliminare un cluster con la protezione dall'eliminazione abilitata, devi modificare il cluster e disabilitare la protezione dall'eliminazione.

Quando si utilizza la console con la protezione dall'eliminazione abilitata su un cluster, non è possibile eliminare l'ultima istanza del cluster perché questa operazione elimina anche il cluster. È possibile eliminare l'ultima istanza di un cluster con protezione dall'eliminazione attiva utilizzando la AWS CLI. Il cluster, tuttavia, è ancora esistente e i dati sono conservati. È possibile accedere ai dati mediante la creazione di nuove istanze per il cluster. Per ulteriori informazioni sull'abilitazione e la disabilitazione della protezione dall'eliminazione, consulta:

- Creazione di un cluster Amazon DocumentDB

- Modifica di un cluster Amazon DocumentDB

Using the Console di gestione AWS

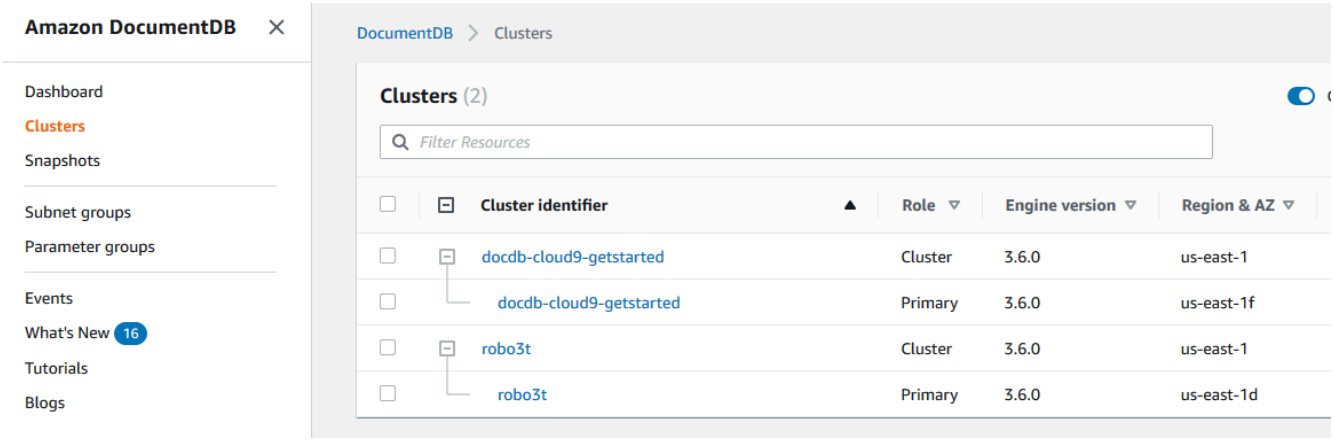
Per eliminare un cluster utilizzando il Console di gestione AWS, la protezione dall'eliminazione deve essere disabilitata.

Per determinare se la protezione dall'eliminazione di un cluster è abilitata:

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.

> **Tip**
>
> Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nota che nella casella di navigazione Clusters, la colonna dell'identificatore del cluster mostra sia i cluster che le istanze. Le istanze sono elencate sotto i cluster, in modo simile alla schermata seguente.



4. Scegliere il nome del cluster e selezionare la scheda Configuration (Configurazione) . Nella sezione Cluster details (Dettagli cluster) individuare l'opzione Deletion protection (Protezione dall'eliminazione). Se la protezione dall'eliminazione è abilitata, modificare il cluster per disabilitarla. Per ulteriori informazioni sulla modifica di un cluster database, consultare Modifica di un cluster Amazon DocumentDB.

Dopo aver eliminato la Deletion protection (Protezione dall'eliminazione), è possibile eliminare il cluster.

Per eliminare un cluster:

1. Nel pannello di navigazione scegliere Cluster.
2. Determina se il cluster ha delle istanze controllando la colonna Identificatore del cluster per le istanze elencate di seguito. Per eliminare un cluster, devi eliminare tutte le sue istanze. Per ulteriori informazioni, consulta Eliminazione di un'istanza Amazon DocumentDB.
3. A seconda che il cluster disponga o meno di qualche istanza, eseguire una delle seguenti operazioni.
   - Se il cluster non dispone di istanze, selezionare il pulsante a sinistra del nome del cluster e scegliere Actions (Operazioni). Dal menu a discesa, scegli Delete (Elimina). Completare la finestra di dialogo Delete <cluster-name> (Elimina <cluster-name>) e quindi scegliere Delete (Elimina).
   - Se il cluster dispone di una o più istanze, eseguire le operazioni descritte qui di seguito:
     a. Nel pannello di navigazione scegliere Cluster.
     b. Elimina ciascuna istanza del cluster selezionando la casella di controllo a sinistra del nome del cluster. Scegli Actions (Operazioni), quindi Delete (Elimina). Completare la finestra di dialogo Delete <cluster-name> (Elimina <cluster-name>) e quindi scegliere Delete (Elimina).

        Quando elimini l'ultima istanza, verrà eliminato anche il cluster. Per ulteriori informazioni sull'eliminazione delle istanze, vedere. Eliminazione di un'istanza Amazon DocumentDB

Sono necessari alcuni minuti per l'eliminazione del cluster. Per monitorare lo stato del cluster, consulta Monitoraggio dello stato di un cluster Amazon DocumentDB.

Using the AWS CLI

Non puoi eliminare un cluster a cui sono associate delle istanze. Per determinare quali istanze sono associate al cluster, esegui il comando `describe-db-clusters` ed elimina tutte le istanze del cluster. Quindi, se necessario, disattiva la protezione dell'eliminazione nel cluster e, infine, elimina il cluster.

1. In primo luogo, elimina tutte le istanze del cluster.

   Per determinare quali istanze eliminare, esegui il comando seguente.

   ```
   aws docdb describe-db-clusters \
       --db-cluster-identifier sample-cluster \
       --query 'DBClusters[*].
   [DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
   ```

   L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

   ```
   [
       [
           "sample-cluster",
           [
               "sample-instance-1",
               "sample-instance-2"
           ]
       ]
   ]
   ```

   Se il cluster da eliminare possiede delle istanze associate, elimina tali istanze prima di continuare.

   ```
   aws docdb delete-db-instance \
       --db-instance-identifier sample-instance
   ```

2. In secondo luogo, disabilita la protezione da eliminazione.

   L'utilizzo AWS CLI di per eliminare tutte le istanze di un cluster non elimina il cluster. È anche necessario eliminare il cluster, ma è possibile eseguire questa operazione solo se la protezione dall'eliminazione è disattivata.

   Per determinare se la protezione dall'eliminazione per il cluster è attivata, eseguire il seguente comando.

> **Tip**
>
> Per visualizzare lo stato di protezione dall'eliminazione di tutti i cluster Amazon DocumentDB, ometti il parametro. `--db-cluster-identifier`

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterIdentifier,DeletionProtection]'
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
[
    [
        "sample-cluster",
        "true"
    ]
]
```

Se la protezione dall'eliminazione per il cluster è abilitata, modificare il cluster e disabilitare la protezione dall'eliminazione. Per disabilitare la protezione dall'eliminazione del cluster, utilizzare il seguente codice.

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --no-deletion-protection \
    --apply-immediately
```

3. Infine, eliminare il cluster.

   Dopo la disattivazione della protezione dall'eliminazione, è possibile eliminare il cluster. Per eliminare un cluster, utilizzare l'operazione `delete-db-cluster` con i parametri elencati di seguito.

   - **`--db-cluster-identifier`**: obbligatorio. L'identificatore del cluster da eliminare.

- **`--final-db-snapshot-identifier`**—Facoltativo. Se desideri uno snapshot finale, devi includere questo parametro con un nome per l'ultimo snapshot. È necessario includere `--final-db-snapshot-identifier` o `--skip-final-snapshot`.

  Vincoli per la denominazione

  - La lunghezza è di [1—63] lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.
  - Deve essere univoco per tutti i cluster di Amazon RDS, Amazon Neptune e Amazon DocumentDB per regione. Account AWS
- **`--skip-final-snapshot`**—Facoltativo. Utilizzare questo parametro solo se non si desidera acquisire uno snapshot finale prima di eliminare il cluster. L'impostazione predefinita prevede l'acquisizione di uno snapshot finale È necessario includere `--final-db-snapshot-identifier` o `--skip-final-snapshot`.

Il AWS CLI codice seguente elimina il cluster `sample-cluster` con un'istantanea finale. L'operazione ha esito negativo se vi sono istanze associate al cluster o se è abilitata la protezione dall'eliminazione.

Example

Per Linux, macOS o Unix:

```
aws docdb delete-db-cluster \
    --db-cluster-identifier sample-cluster \
    --final-db-snapshot-identifier sample-cluster-final-snapshot
```

Per Windows:

```
aws docdb delete-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --final-db-snapshot-identifier sample-cluster-final-snapshot
```

Example

Il AWS CLI codice seguente elimina il cluster `sample-cluster` senza scattare un'istantanea finale.

Per Linux, macOS o Unix:

```
aws docdb delete-db-cluster \
    --db-cluster-identifier sample-cluster \
    --skip-final-snapshot
```

Per Windows:

```
aws docdb delete-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --skip-final-snapshot
```

L'output dell'operazione `delete-db-cluster` è il cluster che stai eliminando.

Sono necessari alcuni minuti per l'eliminazione del cluster. Per monitorare lo stato del cluster, consulta Monitoraggio dello stato di un cluster.

Scalabilità dei cluster Amazon DocumentDB

Amazon DocumentDB ti consente di scalare lo storage e l'elaborazione nei tuoi cluster in base alle tue esigenze. Questa sezione descrive come utilizzare la scalabilità dello storage, la scalabilità delle istanze e la scalabilità in lettura per gestire le prestazioni e la scalabilità dei cluster e delle istanze Amazon DocumentDB.

Argomenti

- Dimensionamento dello storage
- Dimensionamento delle istanze
- Dimensionamento della lettura
- Scalabilità in scrittura

Dimensionamento dello storage

Lo storage di Amazon DocumentDB si ridimensiona automaticamente in base ai dati presenti nel volume del cluster. Man mano che i dati crescono, il volume di storage del cluster aumenta con incrementi di 10 GiB, fino a 256 TiB per la versione 8.0 e successive del motore (128 TiB per le versioni precedenti del motore).

Dimensionamento delle istanze

Puoi scalare il tuo cluster Amazon DocumentDB in base alle esigenze modificando la classe di istanza per ogni istanza del cluster. Amazon DocumentDB supporta diverse classi di istanze ottimizzate per Amazon DocumentDB.

Per ulteriori informazioni, consulta Modifica di un'istanza Amazon DocumentDB.

Dimensionamento della lettura

Puoi ottenere la scalabilità in lettura per il tuo cluster Amazon DocumentDB creando fino a 15 repliche Amazon DocumentDB nel cluster. Ogni replica di Amazon DocumentDB restituisce gli stessi dati dal volume del cluster con un ritardo di replica minimo, in genere meno di 100 millisecondi dopo che l'istanza primaria ha scritto un aggiornamento. Man mano che il traffico di lettura aumenta, puoi creare repliche Amazon DocumentDB aggiuntive e connetterti direttamente ad esse per distribuire il carico di lettura per il tuo cluster. Le repliche di Amazon DocumentDB non devono necessariamente appartenere alla stessa classe di istanza dell'istanza principale.

Per ulteriori informazioni, consulta Aggiungere un'istanza Amazon DocumentDB a un cluster.

Per leggere e scalare con Amazon DocumentDB, connettiti al cluster come set di repliche e distribuisci le letture alle istanze di replica utilizzando le funzionalità di preferenza di lettura integrate del tuo driver. Per ulteriori informazioni, consulta Connessione ad Amazon DocumentDB come set di repliche.

Scalabilità in scrittura

Puoi scalare la capacità di scrittura sul tuo cluster Amazon DocumentDB aumentando le dimensioni dell'istanza primaria del cluster. Questa sezione fornisce due metodi per dimensionare l'istanza primaria del cluster in base alle proprie esigenze. La prima opzione mira a ridurre al minimo l'impatto sull'applicazione, ma richiede più passaggi per essere completata. La seconda opzione ottimizza la semplicità in quanto ha meno fasi, ma ha un impatto potenziale maggiore sull'applicazione.

A seconda dell'applicazione, è possibile scegliere l'approccio più adatto alle proprie esigenze. Per ulteriori informazioni sulle dimensioni e sui costi delle istanze disponibili, consulta la pagina dei prezzi di Amazon DocumentDB.

1. Ottimizzazione per disponibilità e prestazioni elevate: se ti connetti al cluster in modalità set di repliche (consigliata), puoi utilizzare la seguente procedura per ridurre al minimo l'impatto sull'applicazione durante il ridimensionamento dell'istanza principale. Questo metodo riduce al minimo l'impatto perché mantiene il cluster pari o superiore all'elevata disponibilità e le

destinazioni di dimensionamento in lettura vengono aggiunte al cluster come istanze, anziché essere aggiornate.

a. Aggiunta di una o più repliche del tipo di istanza più grande al cluster (consulta ???). Assicurati che tutte le repliche siano dello stesso tipo o di istanza più grande dell'istanza principale. Ciò evita il fileover di una riduzione involontaria delle prestazioni di scrittura su un tipo di istanza più piccolo. Nella maggior parte dei casi, ciò significa raddoppiare temporaneamente il numero di istanze nel cluster e quindi rimuovere le repliche più piccole una volta completato il ridimensionamento.

b. Impostare il livello di failover su tutte le nuove repliche su priorità zero, assicurando che una replica del tipo di istanza più piccolo abbia la priorità di failover più alta. Per ulteriori informazioni, consulta ???.

c. Avviare un failover manuale, che promuoverà una delle nuove repliche come istanza primaria. Per ulteriori informazioni, consulta ???.

> **Note**
>
> Ciò comporterà circa 30 secondi di tempo di inattività per il cluster. Pianifica le attività di conseguenza.

d. Rimuovere tutte le repliche di un tipo di istanza di dimensioni inferiori alla nuova istanza primaria dal cluster.

e. Reimpostare il livello di failover di tutte le istanze sulla stessa priorità (in genere, ciò significa impostarle nuovamente su 1).

Supponiamo, ad esempio, di disporre di un cluster che attualmente contiene tre istanze `r5.large` (una primaria e due repliche) e di voler dimensionare su un tipo di istanza `r5.xlarge`. A tale scopo, è necessario innanzitutto aggiungere al cluster tre istanze di replica `r5.xlarge` e quindi impostare il livello di failover delle nuove repliche `r5.xlarge` su zero. Successivamente, è necessario avviare un failover manuale (tenendo conto che l'applicazione avrà circa 30 secondi di inattività). Una volta completato il failover, è necessario rimuovere dal cluster tutte e tre le istanze `r5.large`, lasciando il cluster dimensionato sulle istanze `r5.xlarge`.

Per aiutare a ottimizzare i costi, le istanze Amazon DocumentDB vengono fatturate in incrementi di un secondo, con un costo minimo di dieci minuti a seguito di una modifica dello stato

fatturabile, ad esempio la creazione, la modifica o l'eliminazione di un'istanza. Per ulteriori informazioni, consulta Ottimizzazione dei costi nella documentazione relativa alle best practice.

2. Ottimizzazione per la semplicità: questo approccio ottimizza per la semplicità. Non espande e non contrae il cluster, ma potrebbe ridurre temporaneamente la capacità di lettura.

   È possibile che la modifica della classe di istanza di una replica comporti che l'istanza non soddisfi le richieste per un breve periodo di tempo, da pochi secondi a meno di 30 secondi. Se ci si connette al cluster in modalità set di repliche (consigliata), la capacità di lettura verrà ridotta di una replica (ad esempio, al 66% della capacità in un cluster a 3 nodi o al 75% in un cluster a 4 nodi, ecc.) durante l'operazione di scalabilità.

   a. Scala una delle istanze di replica nel tuo cluster. Per ulteriori informazioni, consulta Gestione delle classi di istanze.

   b. Attendi che l'istanza sia disponibile (vediMonitoraggio dello stato di un'istanza Amazon DocumentDB).

      > **Note**
      >
      > Ciò comporterà circa 30 secondi di tempo di inattività per il cluster. Pianifica le attività di conseguenza.

   c. Continuate a eseguire i passaggi 1 e 2 finché tutte le istanze delle repliche non sono state scalate, una per una.

   d. Avvia un failover manuale. Ciò promuoverà una delle repliche come istanza principale. Per ulteriori informazioni, consulta Failover di Amazon DocumentDB.

      > **Note**
      >
      > Ciò comporterà fino a 30 secondi di inattività per il cluster, ma spesso richiede meno tempo. Pianifica le attività di conseguenza.

   e. Ridimensiona la precedente istanza primaria (ora una replica).

Clonazione di un volume per un cluster Amazon DocumentDB

Utilizzando la clonazione di Amazon DocumentDB, puoi creare un nuovo cluster che utilizza lo stesso volume del cluster Amazon DocumentDB e contiene gli stessi dati dell'originale. Il processo è

progettato per essere veloce e conveniente. Il nuovo cluster con il relativo volume di dati associato è noto come clone. La creazione di un clone è più veloce ed efficiente in termini di spazio rispetto alla copia fisica dei dati utilizzando una tecnica diversa, ad esempio con il ripristino di uno snapshot.

Amazon DocumentDB supporta la creazione di un clone con provisioning di Amazon DocumentDB da un cluster Amazon DocumentDB con provisioning. Quando si crea un clone utilizzando una configurazione di distribuzione diversa da quella di origine, il clone viene creato utilizzando la versione più recente del motore Amazon DocumentDB della fonte.

Quando crei cloni dai tuoi cluster Amazon DocumentDB, i cloni vengono creati nel tuo AWS account, lo stesso account proprietario del cluster Amazon DocumentDB di origine.

Argomenti

- Panoramica sulla clonazione di Amazon DocumentDB
- Note sull'utilizzo per la clonazione di Amazon DocumentDB
- Come funziona la clonazione di Amazon DocumentDB
- Creazione di un clone di Amazon DocumentDB

Panoramica sulla clonazione di Amazon DocumentDB

Amazon DocumentDB utilizza un protocollo copy-on-write per creare un clone. Questo meccanismo utilizza uno spazio aggiuntivo minimo per creare un clone iniziale. Quando il clone viene creato per la prima volta, Amazon DocumentDB conserva una singola copia dei dati utilizzati dal cluster DB di origine e dal nuovo cluster Amazon DocumentDB (clonato). Lo spazio di archiviazione aggiuntivo viene allocato solo quando vengono apportate modifiche ai dati (sul volume di storage Amazon DocumentDB) dal cluster Amazon DocumentDB di origine o dal clone del cluster Amazon DocumentDB. Per ulteriori informazioni sul protocollo copy-on-write, consulta Come funziona la clonazione di Amazon DocumentDB.

La clonazione di Amazon DocumentDB è particolarmente utile per configurare rapidamente ambienti di test utilizzando i dati di produzione, senza rischiare il danneggiamento dei dati. È possibile utilizzare i cloni per molti tipi di applicazioni di breve durata, ad esempio:

- Sperimenta potenziali cambiamenti (modifiche allo schema e modifiche ai gruppi di parametri, ad esempio) per valutare tutti gli impatti.
- Esegui operazioni che utilizzano in modo intensivo i carichi di lavoro, come l'esportazione di dati o l'esecuzione di query analitiche sul clone.

- Creare una copia del cluster database di produzione per lo sviluppo, il test o altri scopi.

Puoi creare più di un clone dallo stesso cluster Amazon DocumentDB. È anche possibile creare più cloni da un altro clone.

Dopo aver creato un clone di Amazon DocumentDB, puoi configurare le istanze Amazon DocumentDB in modo diverso dal cluster Amazon DocumentDB di origine. Ad esempio, potresti non aver bisogno di un clone per scopi di sviluppo per soddisfare gli stessi requisiti di alta disponibilità del cluster Amazon DocumentDB di produzione di origine. In questo caso, puoi configurare il clone con una singola istanza Amazon DocumentDB anziché con più istanze DB utilizzate dal cluster Amazon DocumentDB.

Una volta terminato di utilizzare il clone per test, sviluppo o altri scopi, è possibile eliminarlo.

Note sull'utilizzo per la clonazione di Amazon DocumentDB

Amazon DocumentDB; la clonazione presenta attualmente le seguenti limitazioni:

- Puoi creare tutti i cloni che desideri, fino al numero massimo di cluster database consentito nella Regione AWS. Tuttavia, dopo aver creato 15 cloni, il clone successivo è una copia completa. L'operazione di clonazione agisce come un ripristino point-in-time.
- Non puoi creare un clone in una AWS regione diversa dal cluster Amazon DocumentDB di origine.
- Non puoi creare un clone da un cluster Amazon DocumentDB privo di istanze DB. Puoi clonare solo cluster Amazon DocumentDB che hanno almeno un'istanza DB.
- Puoi creare un clone in un cloud privato virtuale (VPC) diverso da quello del cluster Amazon DocumentDB. Tuttavia, le sottoreti in quei VPC devono essere associate alle stesse zone di disponibilità.

Come funziona la clonazione di Amazon DocumentDB

La clonazione di Amazon DocumentDB funziona sul livello di storage di un cluster Amazon DocumentDB. Utilizza un protocollo copy-on-write veloce ed efficiente in termini di spazio in termini di supporti durevoli sottostanti che supportano il volume di storage Amazon DocumentDB. Puoi saperne di più sui volumi del cluster Amazon DocumentDB in. Gestione dei cluster Amazon DocumentDB

Argomenti

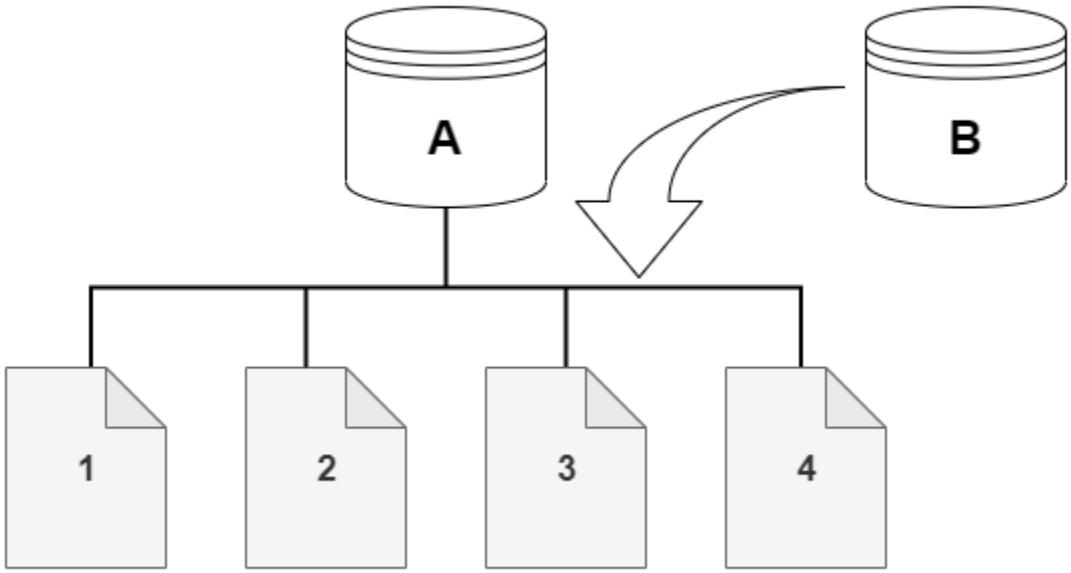
- Informazioni sul protocollo copy-on-write

- Eliminazione di un volume cluster di origine

Informazioni sul protocollo copy-on-write

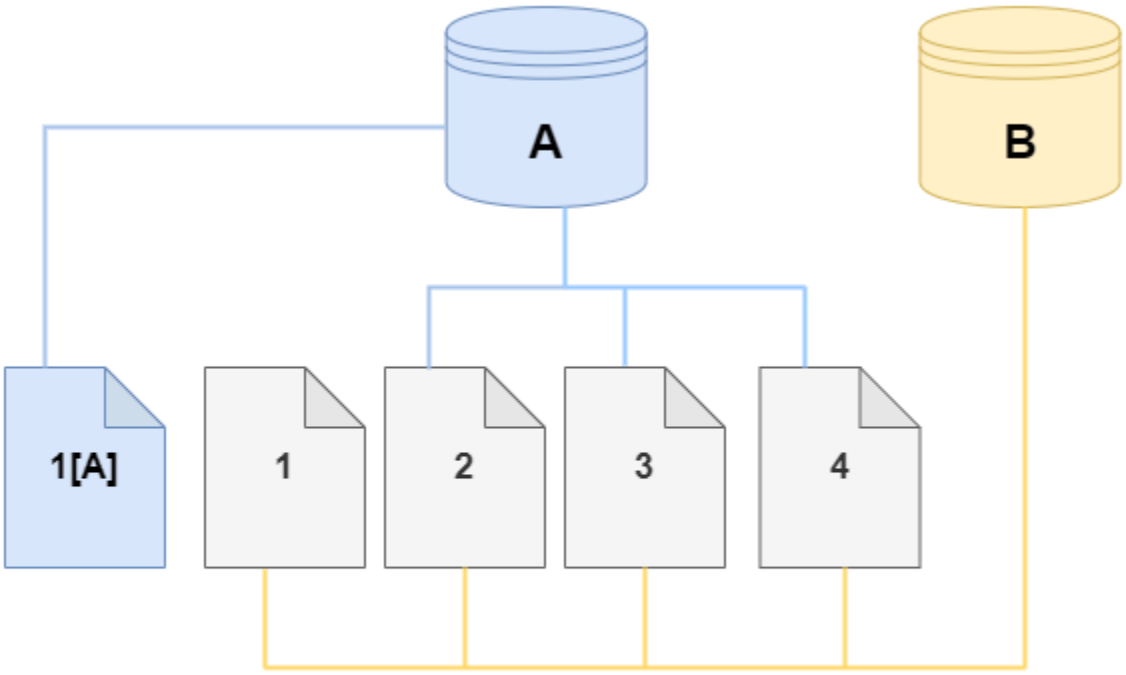
Un cluster Amazon DocumentDB archivia i dati in pagine nel volume di storage Amazon DocumentDB sottostante.

Ad esempio, nel diagramma seguente puoi trovare un cluster Amazon DocumentDB (A) con quattro pagine di dati, 1, 2, 3 e 4. Immagina che un clone, B, venga creato dal cluster Amazon DocumentDB. Quando viene creato il clone, non viene copiato alcun dato. Piuttosto, il clone punta allo stesso set di pagine del cluster Amazon DocumentDB di origine.

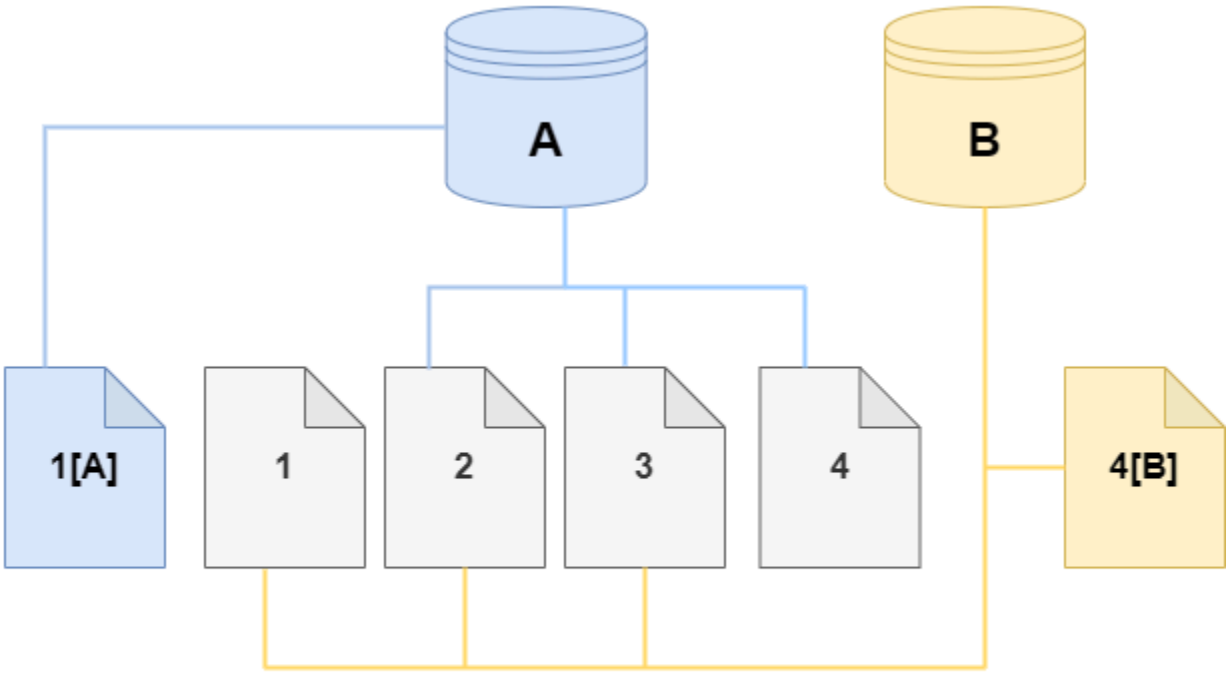


Quando viene creato il clone, in genere non è necessario alcuno spazio di archiviazione aggiuntivo. Il protocollo copy-on-write utilizza lo stesso segmento sul supporto di archiviazione fisico del segmento di origine. Lo spazio di archiviazione aggiuntivo è necessario solo se la capacità del segmento di origine non è sufficiente per l'intero segmento di clone. In questo caso, il segmento di origine viene copiato su un altro dispositivo fisico.

Nei diagrammi seguenti, puoi un esempio del protocollo copy-on-write in azione utilizzando lo stesso cluster A e il relativo clone, B, come illustrato in precedenza. Supponiamo che tu apporti una modifica al tuo cluster Amazon DocumentDB (A) che comporti una modifica ai dati contenuti nella pagina 1. Invece di scrivere sulla pagina 1 originale, Amazon DocumentDB crea una nuova pagina 1 [A]. Il volume del cluster Amazon DocumentDB per il cluster (A) ora punta alle pagine 1 [A], 2, 3 e 4, mentre il clone (B) fa ancora riferimento alle pagine originali.



Sul clone, viene apportata una modifica a pagina 4 sul volume di archiviazione. Invece di scrivere sulla pagina originale 4, Amazon DocumentDB crea una nuova pagina, 4 [B]. Il clone punta ora alle pagine 1, 2, 3 e alla pagina 4[B], mentre il cluster (A) continua a puntare a 1[A], 2, 3 e 4.



Poiché nel tempo si verificano ulteriori modifiche sia nel volume del cluster Amazon DocumentDB di origine che nel clone, è necessario più spazio di archiviazione per acquisire e archiviare le modifiche.

Eliminazione di un volume cluster di origine

Quando si elimina un volume cluster di origine a cui sono associati uno o più cloni, i cloni non sono interessati. I database clone continuano a rimandare alle pagine precedentemente di proprietà del volume del cluster di origine.

Creazione di un clone di Amazon DocumentDB

Puoi creare un clone nello stesso AWS account del cluster Amazon DocumentDB di origine. A tale scopo, puoi utilizzare le procedure Console di gestione AWS o the AWS CLI e le seguenti.

Utilizzando la clonazione di Amazon DocumentDB, puoi creare un clone del cluster Amazon DocumentDB con provisioning da un cluster Amazon DocumentDB con provisioning.

Using the Console di gestione AWS

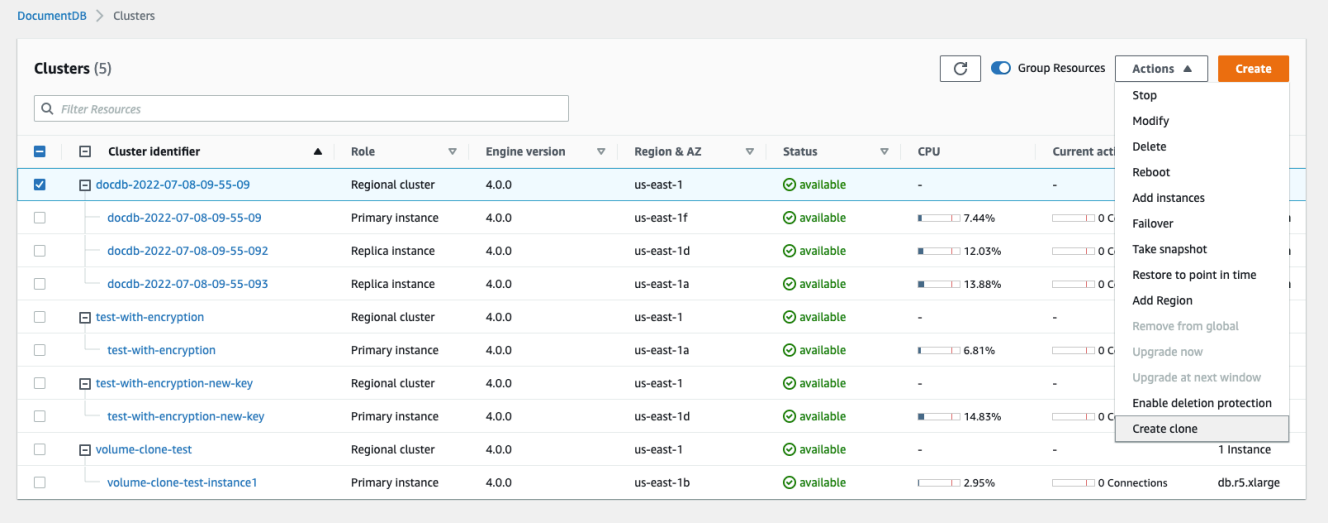
La procedura seguente descrive come clonare un cluster Amazon DocumentDB utilizzando. Console di gestione AWS

Creazione di un clone utilizzando i Console di gestione AWS risultati in un cluster Amazon DocumentDB con un'istanza Amazon DocumentDB.

Queste istruzioni si applicano ai cluster DB di proprietà dello stesso AWS account che crea il clone. Il cluster DB deve essere di proprietà dello stesso AWS account poiché la clonazione tra account non è supportata in Amazon DocumentDB.

Per creare un clone di un cluster DB di proprietà del tuo AWS account utilizzando il Console di gestione AWS

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.
2. Nel pannello di navigazione scegliere Cluster.
3. Scegli il tuo cluster Amazon DocumentDB dall'elenco e, per Azioni, scegli Crea clone.

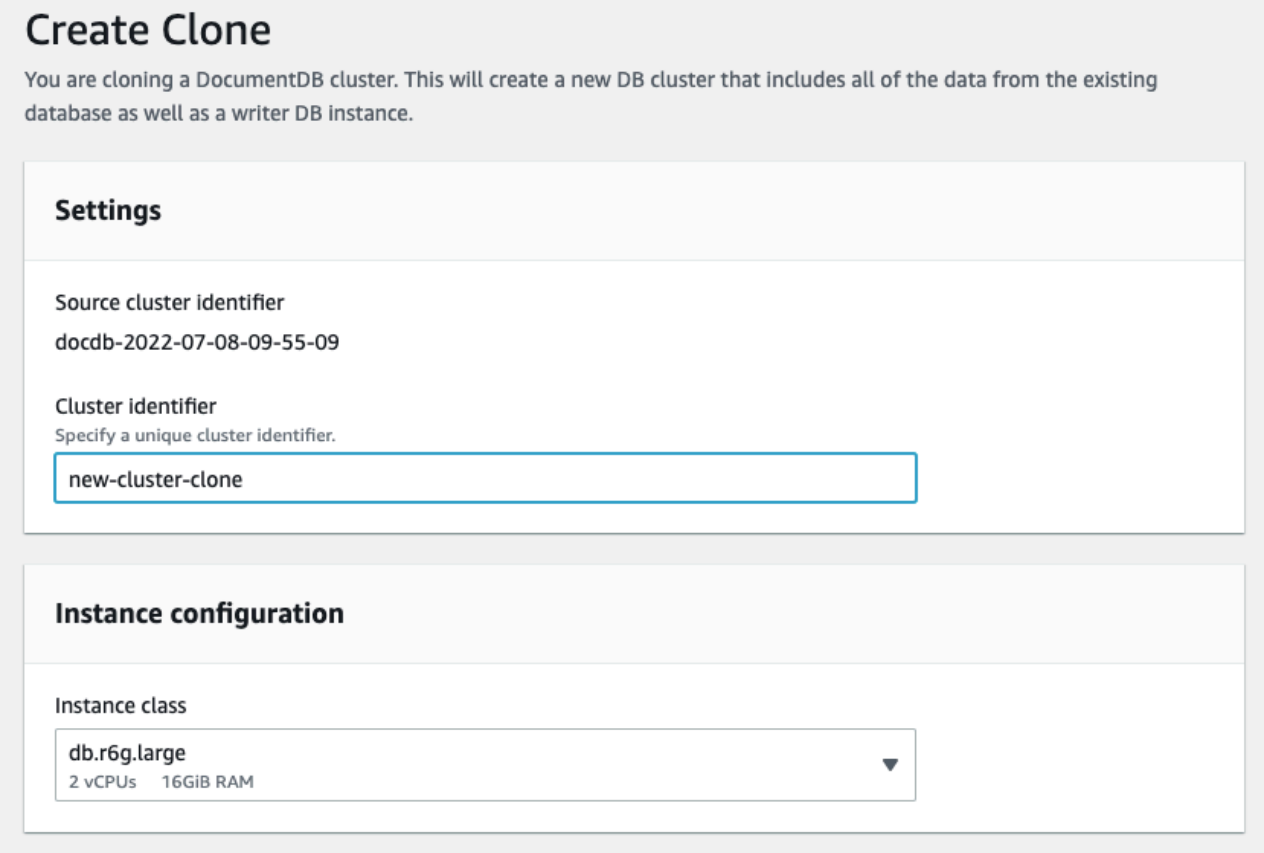


Viene visualizzata la pagina Crea clone, in cui è possibile configurare un identificatore del cluster e una classe di istanza e altre opzioni per il clone del cluster Amazon DocumentDB.

4. Nella sezione Rule settings (Impostazioni regole), procedi nel seguente modo:

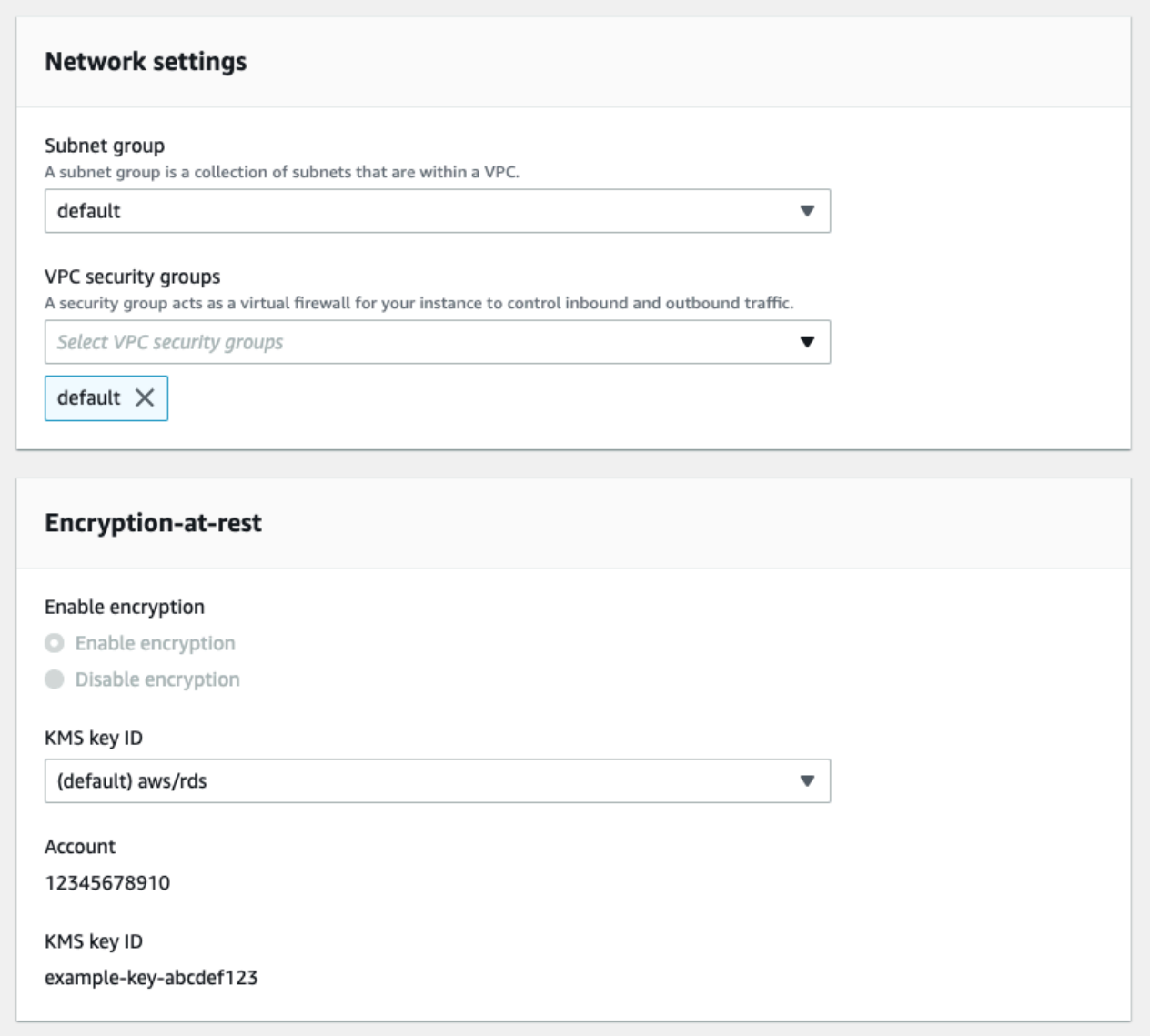
   a. Per l'identificatore del cluster, inserisci il nome che desideri assegnare al cluster Amazon DocumentDB clonato.

b. Per la configurazione dell'istanza, seleziona una classe di istanza appropriata per il cluster Amazon DocumentDB clonato.

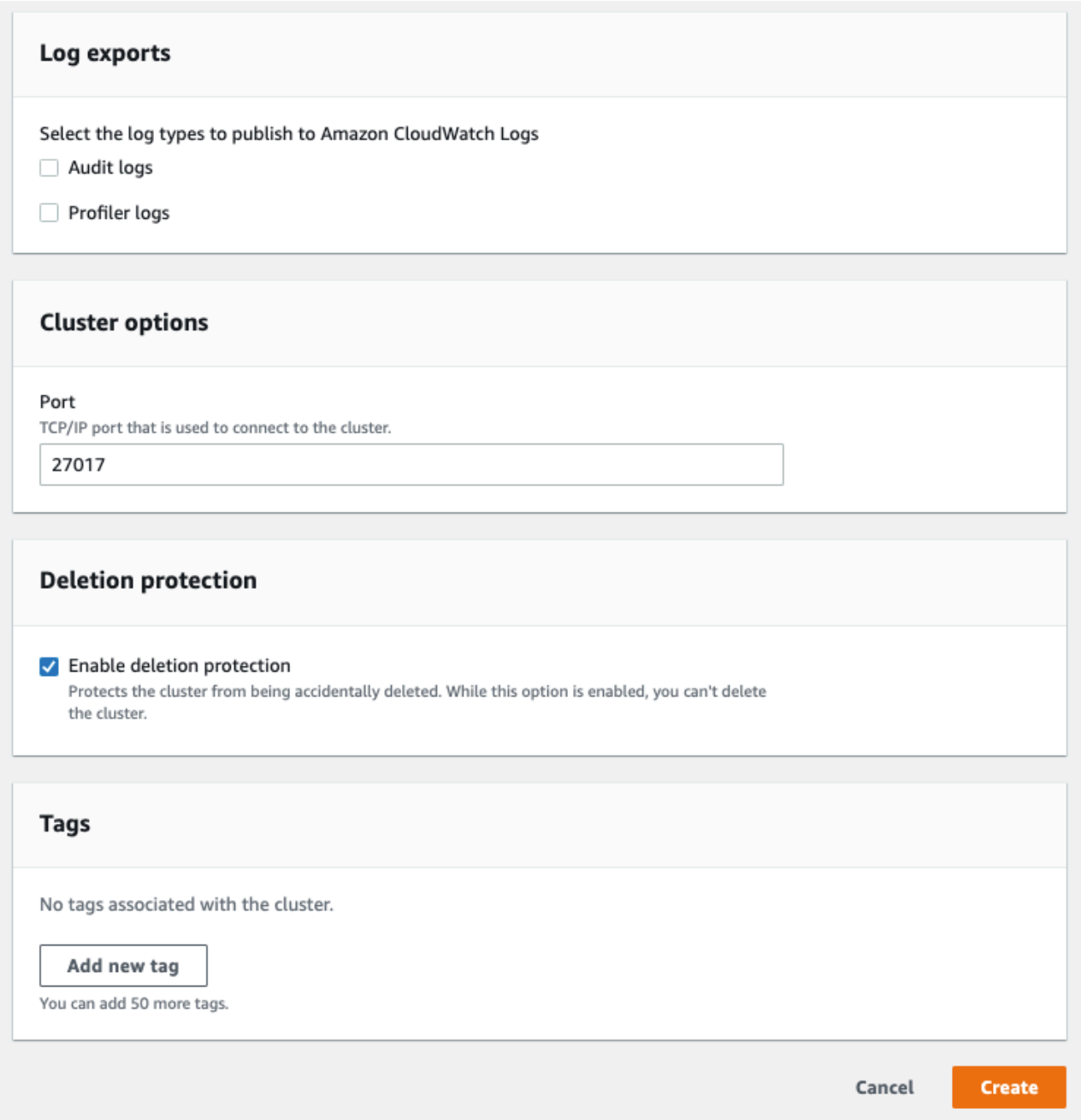


c. Per le impostazioni di rete, scegli un gruppo di sottorete per il tuo caso d'uso e i gruppi di sicurezza VPC associati.

d. Perché Encryption-at-rest, se il cluster di origine (il cluster che viene clonato) ha la crittografia abilitata, anche il cluster clonato deve avere la crittografia abilitata. Se questo scenario è vero, le opzioni Abilita crittografia sono disattivate (disattivate) ma con l'opzione Abilita crittografia selezionata. Al contrario, se nel cluster di origine non è abilitata la crittografia, sono disponibili le opzioni Abilita crittografia ed è possibile scegliere di abilitare o disabilitare la crittografia.



e. Completa la nuova configurazione del clone del cluster selezionando il tipo di log da esportare (opzionale), inserendo una porta specifica utilizzata per la connessione al cluster e abilitando la protezione dall'eliminazione accidentale del cluster (abilitata per impostazione predefinita).



f. Completa l'immissione di tutte le impostazioni per il clone del cluster Amazon DocumentDB. Per ulteriori informazioni sulle impostazioni del cluster e delle istanze Amazon DocumentDB, consulta. Gestione dei cluster Amazon DocumentDB

5. Scegli Crea clone per avviare il clone Amazon DocumentDB del cluster Amazon DocumentDB scelto.

Quando viene creato, il clone viene elencato con gli altri cluster Amazon DocumentDB nella sezione Database della console e ne viene visualizzato lo stato corrente. Il clone è pronto per l'utilizzo quando lo stato diventa Disponibile.

Using the AWS CLI

L'utilizzo di AWS CLI per clonare il cluster Amazon DocumentDB richiede un paio di passaggi.

Il `restore-db-cluster-to-point-in-time` AWS CLI comando utilizzato genera un cluster Amazon DocumentDB vuoto con 0 istanze Amazon DocumentDB. In altre parole, il comando ripristina solo il cluster Amazon DocumentDB, non le istanze DB per quel cluster. Sarà possibile farlo separatamente una volta che il clone è disponibile. Le due fasi del processo sono descritte di seguito:

1. Creare il clone utilizzando il comando restore-db-cluster-to-point-in-time della CLI. I parametri utilizzati con questo comando controllano il tipo di capacità e altri dettagli del cluster Amazon DocumentDB vuoto (clone) che viene creato.
2. Crea l'istanza Amazon DocumentDB per il clone utilizzando il comando CLI create-db-instance per ricreare l'istanza Amazon DocumentDB nel cluster Amazon DocumentDB ripristinato.

I comandi seguenti presuppongono che sia impostato con la AWS CLI tua regione come predefinita. AWS Questo approccio ti evita di dover inviare il nome `--region` in ciascuno dei comandi. Per ulteriori informazioni, consultare la pagina relativa alla configurazione di AWS CLI. È anche possibile specificare la `--region` in ciascuno dei comandi della CLI che seguono.

Creazione del clone

I parametri specifici che vengono inviati al comando della CLI `restore-db-cluster-to-point-in-time` variano. Ciò che si passa dipende dal tipo di clone che si desidera creare.

Utilizza la seguente procedura per creare un clone Amazon DocumentDB con provisioning da un cluster Amazon DocumentDB con provisioning.

Per creare un clone della stessa modalità di motore del cluster Amazon DocumentDB di origine

- Utilizzare il comando `restore-db-cluster-to-point-in-time` della CLI e specificare i valori per i seguenti parametri:
  - `--db-cluster-identifier`: scegliere un nome significativo per il clone. Il nome del clone viene assegnato utilizzando il comando restore-db-cluster-to-point-in-time della CLI.

- `--restore-type`: utilizza `copy-on-write` per creare un clone del cluster database di origine. Senza questo parametro, `restore-db-cluster-to-point-in-time` ripristina il cluster Amazon DocumentDB anziché creare un clone. L'impostazione predefinita per è. `restore-type full-copy`
- `--source-db-cluster-identifier`— Usa il nome del cluster Amazon DocumentDB di origine che desideri clonare.
- `--use-latest-restorable-time`: questo valore punta ai dati del volume ripristinabile più recenti per il clone. Questo parametro è obbligatorio`restore-type copy-on-write`, tuttavia, non è possibile utilizzarlo `restore-to-time parameter` con esso.

Nell'esempio seguente viene creato un clone denominato `my-clone` da un cluster denominato `my-source-cluster`.

Per Linux, macOS o 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
```

Per 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
```

Il comando restituisce l'oggetto JSON contenente i dettagli del clone. Prima di provare a creare l'istanza database per il clone, verificare che il cluster database clonato sia disponibile. Per ulteriori informazioni, consulta Verifica dello stato e acquisizione dei dettagli del clone di seguito:

Verifica dello stato e ottenimento dei dettagli del clone

È possibile utilizzare il seguente comando per controllare lo stato del cluster database vuoto appena creato.

```
$ aws docdb describe-db-clusters --db-cluster-identifier my-clone --query '*[].
[Status]' --output text
```

Oppure puoi ottenere lo stato e gli altri valori necessari per creare l'istanza DB per il tuo clone utilizzando la seguente query: AWS CLI

Per Linux, macOS o Unix:

```
aws docdb describe-db-clusters --db-cluster-identifier my-clone \
  --query '*[].{Status:Status,Engine:Engine,EngineVersion:EngineVersion}'
```

Per Windows:

```
aws docdb describe-db-clusters --db-cluster-identifier my-clone ^
  --query "*[].{Status:Status,Engine:Engine,EngineVersion:EngineVersion}"
```

Questa query restituisce un output simile al seguente:

```
[
  {
        "Status": "available",
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
    }
]
```

Creazione dell'istanza Amazon DocumentDB per il tuo clone

Emettere il comando create-db-instance della CLI per creare l'istanza database per il clone.

Il `--db-instance-class` parametro viene utilizzato solo per i cluster Amazon DocumentDB predisposti.

Per Linux, macOS o 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
```

Per 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
```

Parametri da utilizzare per la clonazione

La tabella seguente riassume i vari parametri utilizzati per `restore-db-cluster-to-point-in-time` clonare i cluster Amazon DocumentDB.

Parametro	Descrizione
--source-db-cluster-identifier	Usa il nome del cluster Amazon DocumentDB di origine che desideri clonare.
--db-cluster-identifier	Scegliere un nome significativo per il clone. È possibile assegnare un nome al clone con il comando `restore-db-cluster-to-point-in-time` . Quindi questo nome viene inviato al comando `create-db-instance` .
--restore-type	Specifica `copy-on-write` come `--restore-type` per creare un clone del cluster DB di origine anziché ripristinare il cluster Amazon DocumentDB di origine.
--use-latest-restorable-time	Questo valore punta ai dati del volume ripristinabile più recenti per il clone.

Comprendere la tolleranza agli errori del cluster Amazon DocumentDB

I cluster Amazon DocumentDB sono tolleranti ai guasti in base alla progettazione. Il volume di ogni cluster si estende su più zone di disponibilità in un'unica Regione AWS zona di disponibilità e ogni zona di disponibilità contiene una copia dei dati del volume del cluster. Questa funzionalità consente al cluster di tollerare il guasto di una zona di disponibilità senza perdita di dati e con solo una breve interruzione del servizio.

Se l'istanza primaria di un cluster fallisce, Amazon DocumentDB esegue automaticamente un failover su una nuova istanza primaria in due modi:

- Promuovendo una replica Amazon DocumentDB esistente nella nuova istanza primaria scelta in base all'impostazione del livello di promozione di ciascuna replica e quindi creando una sostituzione per la precedente primaria. Un failover sull'istanza di replica richiede in genere meno di 30 secondi. Le operazioni di lettura e scrittura possono subire brevi interruzioni durante questo periodo. Per aumentare la disponibilità del cluster, crea almeno una o più repliche Amazon DocumentDB in due o più zone di disponibilità diverse.
- Creando una nuova istanza primaria. Ciò si verifica solo se non è presente un'istanza di replica nel cluster e il completamento può richiedere alcuni minuti.

Se il cluster ha una o più repliche Amazon DocumentDB, una replica Amazon DocumentDB viene promossa all'istanza primaria durante un evento di errore. Un evento di errore ha come conseguenza una breve interruzione, durante la quale le operazioni di lettura e scrittura inviate all'istanza primaria falliscono con un'eccezione. Tuttavia, il servizio viene in genere ripristinato in meno di 120 secondi e spesso in meno di 60 secondi. Per aumentare la disponibilità del cluster, crea almeno una o più repliche Amazon DocumentDB in due o più zone di disponibilità diverse.

Puoi personalizzare l'ordine in cui le tue repliche Amazon DocumentDB vengono promosse all'istanza primaria dopo un errore assegnando a ciascuna replica una priorità. Le priorità vanno da 0, quella massima, a 15, quella minima. Se l'istanza primaria fallisce, la replica Amazon DocumentDB con la priorità più alta viene promossa alla nuova istanza primaria. Puoi modificare la priorità di una replica Amazon DocumentDB in qualsiasi momento. Modificando una priorità non attiverai un failover. Puoi utilizzare l'operazione `modify-db-instance` con il parametro `--promotion-tier`. Per ulteriori informazioni sulla personalizzazione della priorità di failover di un'istanza, consulta Failover di Amazon DocumentDB.

Più di una replica di Amazon DocumentDB può condividere la stessa priorità, generando livelli di promozione. Se due o più repliche di Amazon DocumentDB condividono la stessa priorità, la replica con le dimensioni maggiori viene promossa a primaria. Se due o più repliche Amazon DocumentDB condividono la stessa priorità e dimensione, viene promossa una replica arbitraria nello stesso livello di promozione.

Se il cluster non contiene repliche di Amazon DocumentDB, l'istanza primaria viene ricreata durante un evento di errore. Un evento di errore ha come conseguenza un'interruzione, durante la quale le operazioni di lettura e scrittura falliscono con un'eccezione. Il servizio viene ripristinato quando crei la nuova istanza primaria. In genere, ciò richiede meno di 10 minuti. La promozione di una replica

Amazon DocumentDB sull'istanza primaria è molto più rapida rispetto alla creazione di una nuova istanza primaria.

Gestione delle istanze di Amazon DocumentDB

I seguenti argomenti forniscono informazioni per aiutarti a gestire le istanze di Amazon DocumentDB. Includono dettagli sulle classi e gli stati delle istanze e su come creare, eliminare e modificare un'istanza.

Argomenti

- Determinazione dello stato di un'istanza
- Ciclo di vita delle istanze Amazon DocumentDB
- Gestione delle classi di istanze
- NVMe-backed istanze

Determinazione dello stato di un'istanza

Per visualizzare gli stati validi di un'istanza, il loro significato e come determinare lo stato delle istanze, consulta Monitoraggio dello stato di un'istanza Amazon DocumentDB.

Ciclo di vita delle istanze Amazon DocumentDB

Il ciclo di vita di un'istanza Amazon DocumentDB include la creazione, la modifica, la manutenzione e l'aggiornamento, l'esecuzione di backup e ripristini, il riavvio e l'eliminazione dell'istanza. Questa sezione fornisce informazioni su come completare questi processi.

Argomenti

- Aggiungere un'istanza Amazon DocumentDB a un cluster
- Descrizione delle istanze di Amazon DocumentDB
- Modifica di un'istanza Amazon DocumentDB
- Riavvio di un'istanza Amazon DocumentDB
- Eliminazione di un'istanza Amazon DocumentDB

Puoi creare una nuova istanza di Amazon DocumentDB utilizzando Console di gestione AWS o il. AWS CLI Per aggiungere un'istanza a un cluster, il cluster deve essere nello stato disponibile .

Non è possibile aggiungere un'istanza a un cluster arrestato. Se il cluster viene arrestato, per prima cosa avviare il cluster, attendere che diventi disponibile, quindi aggiungere un'istanza. Per ulteriori informazioni, consulta Arresto e avvio di un cluster Amazon DocumentDB.

> **Note**
>
> Se crei un cluster Amazon DocumentDB utilizzando la console, contemporaneamente viene creata automaticamente un'istanza per te. Per creare altre istanze, utilizza una delle procedure riportate di seguito.

Aggiungere un'istanza Amazon DocumentDB a un cluster

Using the Console di gestione AWS

Utilizza la seguente procedura per creare un'istanza per il tuo cluster utilizzando la console Amazon DocumentDB.

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Per scegliere il cluster a cui aggiungere un'istanza, seleziona il pulsante a sinistra del nome del cluster.
4. Scegli Actions (Operazioni), quindi Add instance (Aggiungi istanza).
5. Nella pagina Add instance to: (Aggiungi istanza a:)<cluster-name> ripetere i seguenti passaggi per ogni istanza che si desidera aggiungere al cluster. Puoi averne fino a 15.

   a. Identificatore dell'istanza: puoi inserire un identificatore univoco per questa istanza o consentire ad Amazon DocumentDB di fornire l'identificatore dell'istanza basato sull'identificatore del cluster.

Vincoli per la denominazione di un'istanza:

- La lunghezza è di [1—63] lettere, numeri o trattini.
- Il primo carattere deve essere una lettera.
- Non può terminare con un trattino o contenere due trattini consecutivi.
- Deve essere unico per tutte le istanze in Amazon RDS, Neptune e Amazon DocumentDB per regione. Account AWS

b. Classe di istanza: dall'elenco a discesa, scegli il tipo di istanza che desideri per questa istanza.

c. Livello di promozione: dall'elenco a discesa, scegli il livello di promozione per la tua istanza o scegli Nessuna preferenza per consentire ad Amazon DocumentDB di impostare il livello di promozione per la tua istanza. Un numero minore indica una priorità più elevata. Per ulteriori informazioni, consulta Controllo della destinazione del failover.

d. Per aggiungere ulteriori istanze, scegliere Add additional instances (Aggiungi istanze aggiuntive) e ripetere i passaggi a, b e c.

6. Terminare l'operazione.

- Per aggiungere istanze al cluster, scegliere Crea.
- Per annullare l'operazione, scegli Cancel (Annulla).

La creazione dell'istanza richiede diversi minuti. Puoi usare la console o AWS CLI visualizzare lo stato dell'istanza. Per ulteriori informazioni, consulta Monitoraggio dello stato di un'istanza.

Using the AWS CLI

Utilizza l'`create-db-instance` AWS CLI operazione con i seguenti parametri per creare l'istanza principale per il tuo cluster.

- **`--db-instance-class`**: obbligatorio. La capacità di calcolo e di memoria dell'istanza, ad esempio `db.m4.large`. Non tutte le classi di istanze sono disponibili in tutte Regioni AWS.
- **`--db-instance-identifier`**: obbligatorio. Una stringa di lettere minuscole che identifica l'istanza.

  Vincoli per la denominazione di un'istanza:

  - La lunghezza è di [1—63] lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.

  - Non può terminare con un trattino o contenere due trattini consecutivi.
  - Deve essere unico per tutte le istanze in Amazon RDS, Neptune e Amazon DocumentDB per regione. Account AWS
- **`--engine`**: obbligatorio. Deve essere `docdb`.
- **`--availability-zone`**— Facoltativo. La zona di disponibilità in cui verrà creata l'istanza. Utilizza questo parametro per trovare le istanze in diverse zone di disponibilità per aumentare la tolleranza ai guasti. Per ulteriori informazioni, consulta Disponibilità e replica elevate di Amazon DocumentDB.
- **`--promotion-tier`**— Facoltativo. Il livello di priorità di failover per l'istanza. Il valore deve essere compreso tra 0 e 15 con i numeri più bassi che corrispondono a una priorità maggiore. Per ulteriori informazioni, consulta Controllo della destinazione del failover.

1. Innanzitutto, determina in quali zone di disponibilità puoi creare l'istanza.

   Se desideri specificare la zona di disponibilità prima di creare l'istanza, esegui il comando seguente per determinare quali zone di disponibilità sono disponibili per il tuo cluster Amazon DocumentDB.

   Per Linux, macOS o Unix:

   ```
   aws docdb describe-db-clusters \
         --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
   ```

   Per Windows:

   ```
   aws docdb describe-db-clusters ^
         --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
   ```

   L'aspetto dell'output di questa operazione è simile al seguente.

   ```
   [
       [
           "sample-cluster",
           [
               "us-east-1c",
               "us-east-1b",
               "us-east-1a"
   ```

```
        ]
    ]
]
```

2. In secondo luogo, determina quali classi di istanza puoi creare nella regione.

   Per determinare le classi di istanze disponibili nella tua regione, esegui il comando seguente. Dall'output, scegli una classe di istanza per l'istanza che desideri aggiungere al tuo cluster Amazon DocumentDB.

   Per Linux, macOS o Unix:

```
aws docdb describe-orderable-db-instance-options \
        --engine docdb \
        --query 'OrderableDBInstanceOptions[*].DBInstanceClass'
```

   Per Windows:

```
aws docdb describe-orderable-db-instance-options ^
        --engine docdb ^
        --query 'OrderableDBInstanceOptions[*].DBInstanceClass'
```

   L'aspetto dell'output di questa operazione è simile al seguente.

```
[
    "db.r5.16xlarge",
    "db.r5.2xlarge",
    "db.r5.4xlarge",
    "db.r5.8xlarge",
    "db.r5.large",
    "db.r5.xlarge"
]
```

3. Infine, aggiungi un'istanza al tuo cluster Amazon DocumentDB.

   Per aggiungere un'istanza al tuo cluster Amazon DocumentDB, esegui il comando seguente.

   Per Linux, macOS o 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
```

Per 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
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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"
    }
}
```

La creazione dell'istanza richiede diversi minuti. Puoi usare la console o AWS CLI visualizzare lo stato dell'istanza. Per ulteriori informazioni, consulta Monitoraggio dello stato di un'istanza Amazon DocumentDB.

Descrizione delle istanze di Amazon DocumentDB

Puoi utilizzare la console di gestione di Amazon DocumentDB o visualizzare dettagli come endpoint di connessione, gruppi di sicurezza VPC, autorità di certificazione e gruppi di parametri relativi alle tue istanze di Amazon DocumentDB. AWS CLI

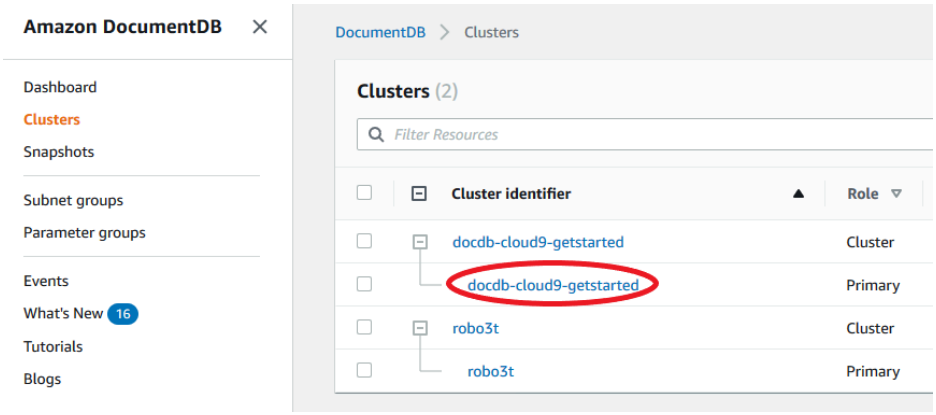
Using the Console di gestione AWS

Per visualizzare i dettagli delle istanze utilizzando la Console di gestione AWS, segui la procedura riportata di seguito.

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.

> **Tip**
>
> Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nella casella di navigazione Clusters, vedrai la colonna Cluster Identifier. Le tue istanze sono elencate in cluster, in modo simile alla schermata seguente.



4. Nell'elenco delle istanze, scegli il nome dell'istanza di cui desideri visualizzarne i dettagli. Le informazioni sull'istanza sono organizzate nei seguenti raggruppamenti:

   - Riepilogo: informazioni generali sull'istanza, tra cui la versione del motore, la classe, lo stato e qualsiasi manutenzione in sospeso.
   - Connettività e sicurezza: la sezione Connect elenca gli endpoint di connessione per connettersi a questa istanza con la shell mongo o con un'applicazione. Nella sezione Security Groups (Gruppi di sicurezza) sono elencati i gruppi di sicurezza associati a questa istanza, l'ID VPC e le descrizioni.

- Configurazione: la sezione Dettagli elenca le configurazioni e lo stato dell'istanza, inclusi l'Amazon Resource Name (ARN), l'endpoint, il ruolo, la classe e l'autorità di certificazione dell'istanza. Vengono inoltre elencate le impostazioni di sicurezza e di rete dell'istanza e le informazioni di backup. La sezione Cluster details (Dettagli cluster) elenca i dettagli del cluster a cui appartiene questa istanza. La sezione Istanze del cluster elenca tutte le istanze che appartengono al cluster con lo stato del ruolo e del gruppo di parametri del cluster di ogni istanza.

  > **Note**
  >
  > Puoi modificare il cluster associato all'istanza selezionando Modify (Modifica) accanto all'intestazione Cluster details (Dettagli cluster) . Per ulteriori informazioni, consulta Modifica di un cluster Amazon DocumentDB.

- Monitoraggio: i parametri dei CloudWatch log per questa istanza. Per ulteriori informazioni, consulta Monitoraggio di Amazon DocumentDB con CloudWatch.
- Eventi e tag: la sezione Eventi recenti elenca gli eventi recenti per questa istanza. Amazon DocumentDB registra gli eventi relativi a cluster, istanze, snapshot, gruppi di sicurezza e gruppi di parametri del cluster. Queste informazioni includono la data, l'ora e il messaggio associati a ciascun evento. La sezione Tag elenca i tag allegati al cluster. Per ulteriori informazioni, consulta Etichettatura delle risorse di Amazon DocumentDB.

Using the AWS CLI

Per visualizzare i dettagli delle istanze di Amazon DocumentDB utilizzando il AWS CLI, usa il `describe-db-clusters` comando come mostrato negli esempi seguenti. Per ulteriori informazioni, consulta `DescribeDBInstances`Amazon DocumentDB Resource Management API Reference.

> **Note**
>
> Per alcune funzionalità di gestione come la gestione del ciclo di vita di cluster e istanze, Amazon DocumentDB sfrutta la tecnologia operativa condivisa con Amazon RDS. Il parametro `filterName=engine,Values=docdb` filter restituisce solo cluster Amazon DocumentDB.

1. Elenca tutte le istanze di Amazon DocumentDB.

   Il AWS CLI codice seguente elenca i dettagli per tutte le istanze di Amazon DocumentDB in una regione.

   Per Linux, macOS o Unix:

   ```
   aws docdb describe-db-instances \
       --filter Name=engine,Values=docdb
   ```

   Per Windows:

   ```
   aws docdb describe-db-instances \
       --filter Name=engine,Values=docdb
   ```

2. Elenca tutti i dettagli per un'istanza Amazon DocumentDB specificata

   Il codice seguente consente di elencare i dettagli per `sample-cluster-instance`. Se includi il parametro `--db-instance-identifier` nel nome di un'istanza, puoi limitare l'output alle sole informazioni su quella specifica istanza.

   Per Linux, macOS o Unix:

   ```
   aws docdb describe-db-instances \
       --db-instance-identifier sample-cluster-instance
   ```

   Per Windows:

   ```
   aws docdb describe-db-instances \
       --db-instance-identifier sample-cluster-instance
   ```

   L'aspetto dell'output di questa operazione è simile al seguente.

   ```
   {
       "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"
        }
    ]
}
```

Modifica di un'istanza Amazon DocumentDB

Puoi modificare la tua istanza di Amazon DocumentDB utilizzando Console di gestione AWS o. AWS CLI Per modificare un'istanza, questa deve essere nello stato disponibile . Non è possibile modificare un'istanza arrestata. Se il cluster viene arrestato, per prima cosa avviare il cluster, attendere che l'istanza diventi disponibile, quindi apportare le modifiche desiderate. Per ulteriori informazioni, consulta Arresto e avvio di un cluster Amazon DocumentDB.

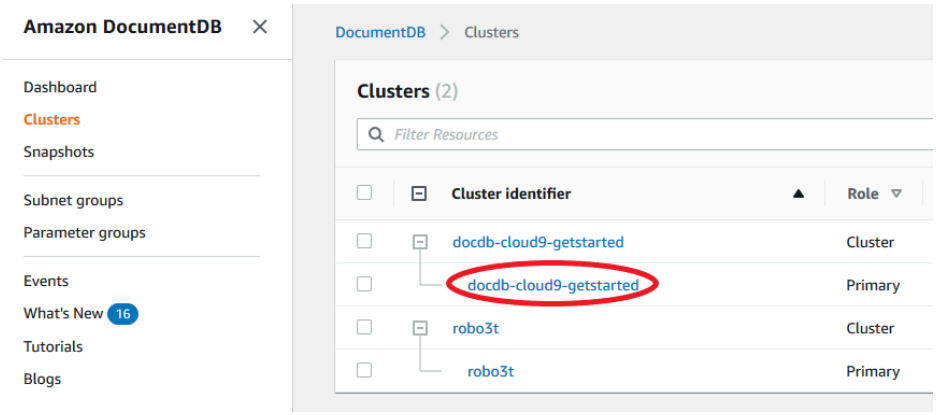
Using the Console di gestione AWS

Per modificare un'istanza specifica di Amazon DocumentDB utilizzando la console, completa i seguenti passaggi.

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.

> **Tip**
>
> Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nella casella di navigazione Clusters, vedrai la colonna Cluster Identifier. Le tue istanze sono elencate in cluster, in modo simile alla schermata seguente.



4. Seleziona la casella a sinistra dell'istanza che desideri modificare.
5. Scegli Actions (Operazioni), quindi Modify (Modifica).
6. Nel riquadro Modify instance: <instance-name>(Modifica istanza: instance-name) apporta le modifiche desiderate. È possibile apportare le seguenti modifiche.

   - Specifiche dell'istanza: l'identificatore e la classe dell'istanza. Vincoli di denominazione dell'identificatore di istanza:
     - Identificatore dell'istanza: inserisci un nome univoco per tutte le istanze di tua proprietà Account AWS nella regione corrente. L'identificatore di istanza deve contenere [1—

63] caratteri alfanumerici o trattini, avere una lettera come primo carattere e non può terminare con un trattino o contenere due trattini consecutivi.

- Classe di istanza: dal menu a discesa, seleziona una classe di istanza per la tua istanza Amazon DocumentDB. Per ulteriori informazioni, consulta Gestione delle classi di istanze.

- Autorità di certificazione: certificato del server per questa istanza. Per ulteriori informazioni, consulta Aggiornamento dei certificati TLS di Amazon DocumentDB.
- Failover: durante il failover, l'istanza con il livello di promozione più alto verrà promossa a principale. Per ulteriori informazioni, consulta Failover di Amazon DocumentDB.
- Manutenzione: la finestra di manutenzione in cui le modifiche o le patch in sospeso vengono applicate alle istanze del cluster.

7. Al termine, scegli Continue (Continua) per visualizzare un riepilogo delle modifiche.
8. Dopo aver verificato le modifiche, è possibile applicarle immediatamente o durante la successiva finestra di manutenzione in Scheduling of modifications (Pianificazione delle modifiche). Scegli Instance (Modifica istanza) per salvare le modifiche. In alternativa, puoi selezionare Cancel (Annulla) per eliminare le modifiche.

L'applicazione delle modifiche richiede qualche minuto. Puoi utilizzare l'istanza solo quando ha lo stato disponibile. Puoi monitorare lo stato dell'istanza con la console o l' AWS CLI. Per ulteriori informazioni, consulta Monitoraggio dello stato di un'istanza Amazon DocumentDB.

Using the AWS CLI

Per modificare un'istanza specifica di Amazon DocumentDB utilizzando il AWS CLI, utilizza `modify-db-instance` i seguenti parametri. Per ulteriori informazioni, vedete ModifyDbInstance. Il codice seguente modifica la classe di istanza in `db.r5.large` per l'istanza `sample-instance`.

Parameters

- **`--db-instance-identifier`**: obbligatorio. L'identificatore dell'istanza da modificare.
- **`--db-instance-class`**— Facoltativo. La nuova capacità di calcolo e memoria dell'istanza, `db.r5.large` ad esempio. Non tutte le classi di istanze sono disponibili in tutte Regioni AWS. Se si modifica la classe dell'istanza, si verifica un'interruzione durante la modifica. La modifica viene applicata durante la finestra di manutenzione successiva, a meno che non `ApplyImmediately` sia specificata come true per questa richiesta.

- **`--apply-immediately`**o **`--no-apply-immediately`** — Facoltativo. Specifica se questa modifica debba essere applicata subito o alla prossima finestra di manutenzione. Se questo parametro viene omesso, la modifica viene eseguita durante la finestra di manutenzione successiva.

Example

Per Linux, macOS o Unix:

```
aws docdb modify-db-instance \
       --db-instance-identifier sample-instance \
       --db-instance-class db.r5.large \
       --apply-immediately
```

Per Windows:

```
aws docdb modify-db-instance ^
       --db-instance-identifier sample-instance ^
       --db-instance-class db.r5.large ^
       --apply-immediately
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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"
```

```
            ]
        }
    ]
}
```

L'applicazione delle modifiche richiede qualche minuto. Puoi utilizzare l'istanza solo quando ha lo stato disponibile. È possibile monitorare lo stato dell'istanza utilizzando Console di gestione AWS o AWS CLI. Per ulteriori informazioni, consulta Monitoraggio dello stato di un'istanza Amazon DocumentDB.

Riavvio di un'istanza Amazon DocumentDB

Occasionalmente, potrebbe essere necessario riavviare l'istanza di Amazon DocumentDB, di solito per motivi di manutenzione. Se apporti alcune modifiche, ad esempio la modifica del gruppo di parametri del cluster associato a un cluster, devi riavviare le istanze del cluster affinché le modifiche abbiano effetto. È possibile riavviare un'istanza specificata utilizzando o il Console di gestione AWS . AWS CLI

Il riavvio di un'istanza comporta il riavvio del servizio del motore di database. Il riavvio comporta un'interruzione temporanea, durante la quale lo stato dell'istanza viene impostato `rebooting`. Un evento Amazon DocumentDB viene creato al termine del riavvio.

Il riavvio di un'istanza non comporta un failover. Per eseguire il failover di un cluster Amazon DocumentDB, usa Console di gestione AWS l'operazione o AWS CLI . `failover-db-cluster` Per ulteriori informazioni, consulta Failover di Amazon DocumentDB.

Non puoi riavviare l'istanza se il suo stato non è disponibile. Il database può non essere disponibile per diversi motivi, ad esempio una modifica richiesta in precedenza o un'operazione della finestra di manutenzione. Per ulteriori informazioni sugli stati delle istanze, consulta Monitoraggio dello stato di un'istanza Amazon DocumentDB.

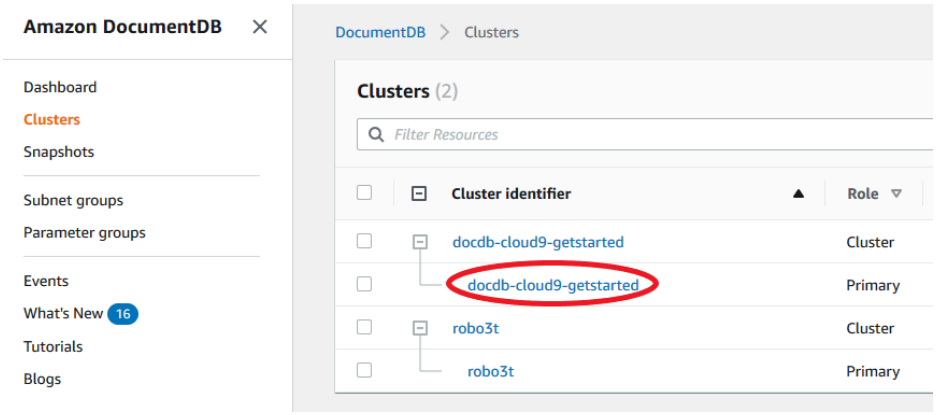
Using the Console di gestione AWS

La procedura seguente riavvia un'istanza specificata utilizzando la console.

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.

> **Tip**
>
> Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nella casella di navigazione Clusters, vedrai la colonna Cluster Identifier. Le tue istanze sono elencate in cluster, in modo simile alla schermata seguente.



4. Seleziona la casella a sinistra dell'istanza che desideri riavviare.
5. Scegli Actions (Operazioni), scegli Reboot (Riavvia) e quindi Reboot (Riavvia) per confermare il riavvio.

Per il riavvio dell'istanza sono necessari alcuni minuti. Puoi utilizzare l'istanza solo quando ha lo stato disponibile. Puoi monitorare lo stato dell'istanza con la console o l' AWS CLI. Per ulteriori informazioni, consulta Monitoraggio dello stato di un'istanza Amazon DocumentDB.

Using the AWS CLI

Per riavviare un'istanza Amazon DocumentDB, utilizza l'operazione con `reboot-db-instance` `--db-instance-identifier` il parametro. Questo parametro specifica l'identificatore per l'istanza da riavviare.

Il codice seguente riavvia l'istanza `sample-instance`.

Example

Per Linux, macOS o Unix:

```
aws docdb reboot-db-instance \
       --db-instance-identifier sample-instance
```

Per Windows:

```
aws docdb reboot-db-instance ^
       --db-instance-identifier sample-instance
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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"
        ]
    }
}
```

Per il riavvio dell'istanza sono necessari alcuni minuti. Puoi utilizzare l'istanza solo quando ha lo stato disponibile. Puoi monitorare lo stato dell'istanza con la console o l' AWS CLI. Per ulteriori informazioni, consulta Monitoraggio dello stato di un'istanza Amazon DocumentDB.

Eliminazione di un'istanza Amazon DocumentDB

Puoi eliminare la tua istanza di Amazon DocumentDB utilizzando Console di gestione AWS o. AWS CLI Per eliminare un'istanza, questa deve essere nello stato disponibile. Non è possibile eliminare un'istanza arrestata. Se il cluster Amazon DocumentDB che contiene l'istanza viene interrotto, avvia prima il cluster, attendi che l'istanza diventi disponibile, quindi elimina l'istanza. Per ulteriori informazioni, consulta Arresto e avvio di un cluster Amazon DocumentDB.

> **Note**
>
> Amazon DocumentDB archivia tutti i dati nel volume del cluster. I dati persistono nel volume cluster, anche se rimuovi tutte le istanze dal cluster. Se hai bisogno di accedere nuovamente ai dati, puoi aggiungere un'istanza al cluster in qualsiasi momento e riprendere da dove avevi lasciato.

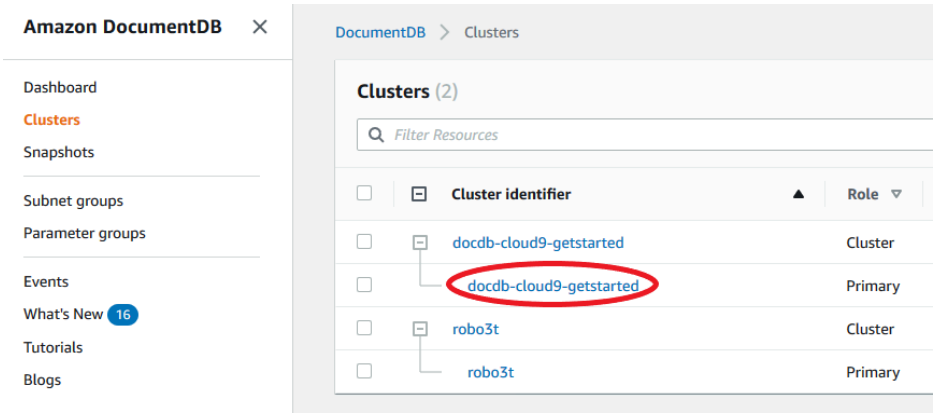
Using the Console di gestione AWS

La procedura seguente elimina un'istanza Amazon DocumentDB specificata utilizzando la console.

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nella casella di navigazione Clusters, vedrai la colonna Cluster Identifier. Le tue istanze sono elencate in cluster, in modo simile alla schermata seguente.



4. Seleziona la casella a sinistra dell'istanza che desideri eliminare.
5. Scegli Actions (Operazioni), quindi Delete (Elimina).

1. Se stai eliminando l'ultima istanza nel cluster:
   - Create final cluster snapshot? (Crea la snapshot del cluster finale?) — Scegliete Sì se desiderate creare un'istantanea finale prima dell'eliminazione del cluster. Altrimenti, scegli No.
   - Nome dell'istantanea finale: se scegli di creare un'istantanea finale, inserisci l'identificatore dell'istantanea del cluster della nuova istantanea del cluster creata.
   - Delete <instance-name> instance? (Elimina istanza <nome-istanza>?) — Inserisci la frase delete entire cluster nel campo per confermare l'eliminazione.
2. Se non stai eliminando l'ultima istanza nel cluster:
   - Delete <instance-name> instance? (Elimina istanza <nome-istanza>?) — Inserisci la frase delete me nel campo per confermare l'eliminazione.

6. Seleziona Delete (Elimina) per eliminare l'istanza.

Sono necessari alcuni minuti per l'eliminazione dell'istanza. Per monitorare lo stato di un'istanza, consulta Monitoraggio dello stato di un'istanza Amazon DocumentDB.

Using the AWS CLI

La procedura seguente elimina un'istanza di Amazon DocumentDB utilizzando. AWS CLI

1. Innanzitutto, determina quante istanze ci sono nel tuo cluster Amazon DocumentDB:

   Per determinare il numero di istanze presenti nel cluster, esegui il comando `describe-db-clusters` come indicato di seguito.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].
[DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
```

   L'aspetto dell'output di questa operazione è simile al seguente.

```
[
    [
        "sample-cluster",
        [
            "sample-instance-1",
            "sample-instance-2"
```

```
        ]
    ]
]
```

2. Se nel cluster Amazon DocumentDB sono presenti più di un'istanza:

   Per eliminare un'istanza Amazon DocumentDB specificata, usa il `delete-db-instance` comando con il `--db-instance-identifier` parametro, come illustrato di seguito. Sono necessari alcuni minuti per l'eliminazione dell'istanza. Per monitorare lo stato di un'istanza, consulta Monitoraggio dello stato di un'istanza Amazon DocumentDB.

   ```
   aws docdb delete-db-instance \
         --db-instance-identifier sample-instance-2
   ```

   L'aspetto dell'output di questa operazione è simile al seguente.

   ```
   {
       "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. Se l'istanza che desideri eliminare è l'ultima istanza del tuo cluster Amazon DocumentDB:

   Se elimini l'ultima istanza in un cluster Amazon DocumentDB, elimini anche quel cluster e gli snapshot automatici e i backup continui associati a quel cluster.

Per eliminare l'ultima istanza del cluster, puoi eliminare il cluster e, facoltativamente, creare uno snapshot finale. Per ulteriori informazioni, consulta Eliminazione di un cluster Amazon DocumentDB.

Protezione da eliminazione

L'eliminazione dell'ultima istanza di un cluster Amazon DocumentDB comporta anche l'eliminazione del cluster, nonché gli snapshot automatici e i backup continui associati a tale cluster. Amazon DocumentDB applica la protezione da eliminazione per un cluster indipendentemente dal fatto che l'operazione di eliminazione venga eseguita utilizzando o il Console di gestione AWS . AWS CLI Se la protezione dall'eliminazione è abilitata, non puoi eliminare un cluster.

Per eliminare un cluster con la protezione dall'eliminazione abilitata, devi modificare il cluster e disabilitare la protezione dall'eliminazione. Per ulteriori informazioni, consulta Eliminazione di un cluster Amazon DocumentDB.

Gestione delle classi di istanze

La classe di istanza determina la capacità di calcolo e di memoria di un'istanza Amazon DocumentDB (con compatibilità MongoDB). La classe di istanza di cui hai bisogno dipende dai requisiti in termini di potenza di elaborazione e memoria.

Amazon DocumentDB supporta le famiglie di classi di istanze R4, R5, R6G, R8G, T3 e T4G. ovvero classi di istanze della generazione attuale ottimizzate per applicazioni a elevato utilizzo di memoria. Per le specifiche relative a tali classi, consulta Specifiche della classe di istanza.

Argomenti

- Determinare una classe di istanza
- Modifica della classe di un'istanza
- Classi di istanze supportate per regione
- Specifiche della classe di istanza

Determinare una classe di istanza

Per determinare la classe di un'istanza, è possibile utilizzare l'`describe-db-instances` AWS CLI operazione Console di gestione AWS o the.

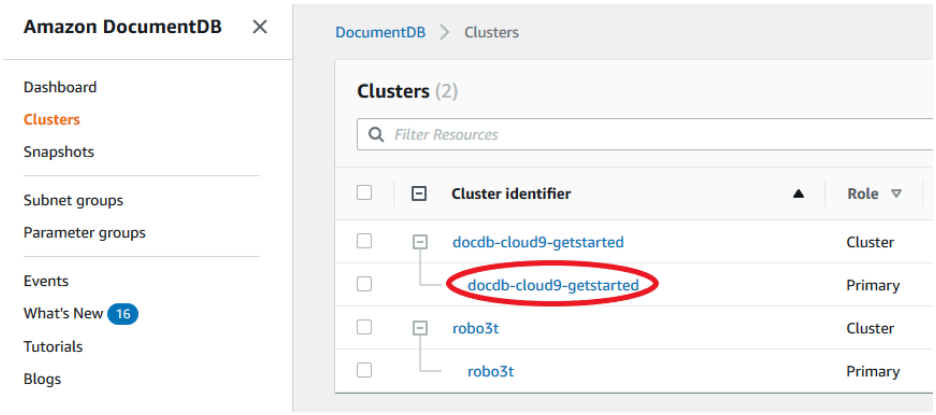
Using the Console di gestione AWS

Per determinare la classe di istanza per le istanze del cluster, completa i seguenti passaggi nella console.

1. Accedi e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione, scegli Clusters per trovare l'istanza che ti interessa.

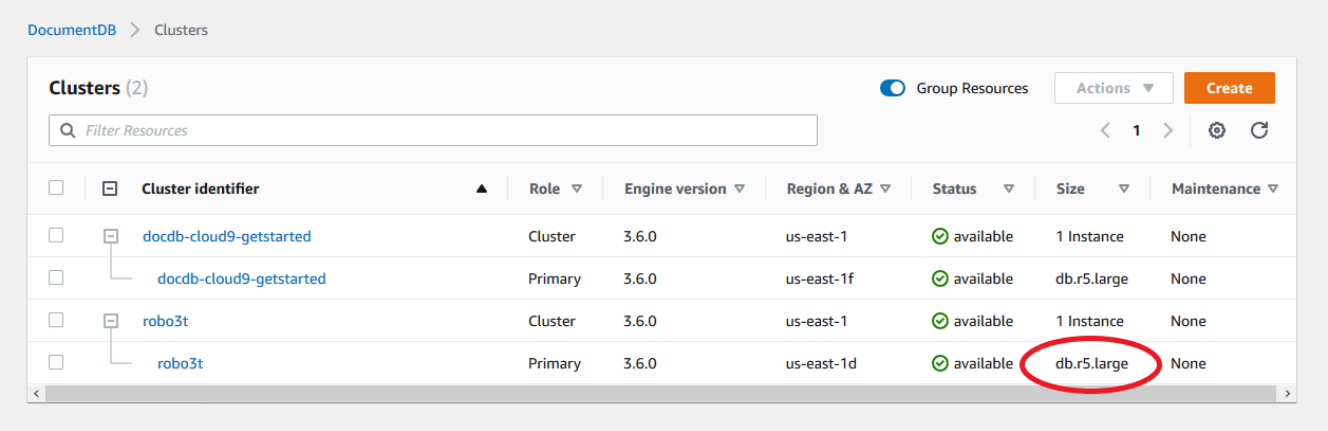
> **Tip**
>
> Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nella casella di navigazione Clusters, vedrai la colonna Cluster Identifier. Le tue istanze sono elencate in cluster, in modo simile alla schermata seguente.



4. Nell'elenco delle istanze, espandi il cluster per trovare le istanze che ti interessano. Trova l'istanza che desideri. Quindi, guarda la colonna Size della riga dell'istanza per vedere la relativa classe di istanza.

   Nell'immagine seguente, la classe per l'istanza `robo3t` è `db.r5.4xlarge`.



Using the AWS CLI

Per determinare la classe di un'istanza utilizzando il AWS CLI, utilizzate l'`describe-db-instances`operazione con i seguenti parametri.

- **`--db-instance-identifier`** : opzionale. Specifica l'istanza della quale desideri trovare la classe. Se questo parametro viene omesso, `describe-db-instances` restituisce una descrizione per un massimo di 100 istanze.
- **`--query`** : opzionale. Specifica i membri dell'istanza da includere nei risultati. Se questo parametro viene omesso, vengono restituiti tutti i membri dell'istanza.

Example

L'esempio seguente trova il nome e la classe dell'istanza`sample-instance-1`.

Per Linux, macOS o Unix:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' \
    --db-instance-identifier sample-instance-1
```

Per Windows:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' ^
    --db-instance-identifier sample-instance-1
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
[
    [
        "sample-instance-1",
        "db.r5.large"
    ]
```

Example

L'esempio seguente trova il nome e la classe dell'istanza per un massimo di 100 istanze Amazon DocumentDB.

Per Linux, macOS o Unix:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' \
    --filter Name=engine,Values=docdb
```

Per Windows:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' ^
    --filter Name=engine,Values=docdb
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
[
    [
        "sample-instance-1",
        "db.r5.large"
    ],
    [
        "sample-instance-2",
        "db.r5.large"
    ],
    [
        "sample-instance-3",
        "db.r5.4xlarge"
    ],
    [
        "sample-instance-4",
        "db.r5.4xlarge"
    ]
```

```
]
```

Per ulteriori informazioni, consulta Descrizione delle istanze di Amazon DocumentDB.

Modifica della classe di un'istanza

È possibile modificare la classe di istanza della propria istanza utilizzando Console di gestione AWS o AWS CLI. Per ulteriori informazioni, consulta Modifica di un'istanza Amazon DocumentDB.

Classi di istanze supportate per regione

Amazon DocumentDB supporta le seguenti classi di istanze:

- R8G—Istanze ottimizzate per la memoria di ultima generazione basate su processori Arm-based AWS Graviton4 che forniscono prestazioni fino al 30% migliori rispetto alle istanze R6G.
- R6G— Memory-optimized istanze basate su processori Arm-based AWS Graviton2 che offrono prestazioni fino al 30% migliori rispetto alle istanze R5 a un costo inferiore del 5%.
- R6GD— istanze Memory-optimized R6G con storage locale basato su unità SSD (NVMe) non volatile memory express (NVMe) per dati temporanei. Solid-State
- R5— Memory-optimized istanze che offrono prestazioni fino al 100% migliori rispetto alle istanze R4 a parità di costo.
- R4—Generazione precedente di istanze ottimizzate per la memoria.
- T4G— tipo di istanza multiuso burstable Latest-generation a basso costo alimentata da processori Arm-based AWS Graviton2 che fornisce un livello base di prestazioni della CPU, offrendo un rapporto qualità-prezzo fino al 35% migliore rispetto alle istanze T3 e ideale per l'esecuzione di applicazioni con un utilizzo moderato della CPU che subiscono picchi temporanei di utilizzo.
- T3—Tipo di istanza multiuso a basso costo e stabile che fornisce un livello base di prestazioni della CPU con la possibilità di aumentare l'utilizzo della CPU in qualsiasi momento per tutto il tempo necessario.

Per specifiche dettagliate relative alle classi di istanze, consulta Specifiche della classe di istanza.

> **Note**
>
> I/O-Optimized lo storage è disponibile solo nelle versioni del motore 5.0 e 8.0 (cluster basati su istanze).

Una particolare classe di istanza può essere supportata o meno in una determinata regione. La tabella seguente specifica quali classi di istanze sono supportate da Amazon DocumentDB in ciascuna regione.

Classi di istanze supportate per regione

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Serverless
Stati Uniti orientali (Ohio)	Supportata	Supportato	Supportato	Supportato	Supportato	Supportato	Supportato	Supportata
Stati Uniti orientali (Virginia settentrionale)	Supportata	Supportato	Supportato	Supportato	Supportato	Supportato	Supportato	Supportata
Stati Uniti occidentali (Oregon)	Supportata	Supportato	Supportato	Supportato	Supportato	Supportato	Supportato	Supportata
Africa (Città del Capo)			Supportata	Supportato		Supportato	Supportato	Supportata
Sud America (San Paolo)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Hong Kong)	Supportata		Supportato	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Hyderabad)			Supportata	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Malesia)			Supportata			Supportato	Supportato	Supportata
Asia Pacifico (Mumbai)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Osaka)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Serverless
Asia Pacifico (Seul)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Sydney)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Giacarta)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Melbourne)			Supportata	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Singapore)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Asia Pacifico (Thailandia)			Supportata			Supportato	Supportato	Supportata
Asia Pacifico (Tokyo)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Canada (Centrale)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Canada occidentale (Calgary)			Supportata	Supportato		Supportato	Supportato	Supportata
Europa (Francoforte)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Europa (Zurigo)		Supportata	Supportato	Supportato		Supportato	Supportato	Supportata
Europa (Irlanda)	Supportata	Supportato	Supportato	Supportato	Supportato	Supportato	Supportato	Supportata

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Serverless
Europa (Londra)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Europa (Milano)			Supportata	Supportato		Supportato	Supportato	Supportata
Europa (Parigi)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Europa (Spagna)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Europa (Stoccolma)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Messico (centrale)			Supportata			Supportato	Supportato	Supportata
Medio Oriente (Emirati Arabi Uniti)			Supportata	Supportato		Supportato	Supportato	Supportata
Cina (Pechino)	Supportata	Supportato	Supportato	Supportato		Supportato	Supportato	Supportata
Cina (Ningxia)	Supportata		Supportato	Supportato		Supportato	Supportato	Supportata
Israele (Tel Aviv)			Supportata	Supportato		Supportato	Supportato	Supportato
AWS GovCloud (US-West)	Supportato	Supportato	Supportato	Supportato			Supportato	Supportato
AWS GovCloud (US-East)		Supportato	Supportato	Supportato		Supportato	Supportato	Supportata

Specifiche della classe di istanza

La tabella seguente fornisce dettagli sulle classi di istanze Amazon DocumentDB, inclusi i tipi di istanze supportati in ciascuna classe. Puoi trovare spiegazioni su ogni colonna della tabella nella sezione sotto la tabella.

Classe di istanza	vCPU[1]	Memoria (GiB) 2	Cache SSD NVMe a più livelli (GiB) 3	Temperatura massima di archiviazione (GiB) 4	Larghezza di banda basale/burst (Gbps) 5	Motori di supporto 6
R8G — Classe di Memory-Optimized istanza di generazione attuale basata su Graviton4 R8G è la famiglia di istanze più recente (Graviton4), disponibile solo nelle versioni del motore 5.0 e 8.0.						
`db.r8g.large`	2	16	-	30	0,937/12,5	5.0.0 e 8.0.0
`db.r8g.xlarge`	4	32	-	60	1,875/12,5	5.0.0 e 8.0.0
`db.r8g.2xlarge`	8	64	-	121	3,75/15,0	5.0.0 e 8.0.0
`db.r8g.4xlarge`	16	128	-	243	7,5/15,0	5.0.0 e 8.0.0
`db.r8g.8xlarge`	32	256	-	488	15	5.0.0 e 8.0.0
`db.r8g.12xlarge`	48	384	-	732	22	5.0.0 e 8.0.0
`db.r8g.16xlarge`	64	512	-	987	30	5.0.0 e 8.0.0

Classe di istanza	vCPU[1]	Memoria (GiB) 2	Cache SSD NVMe a più livelli (GiB) 3	Temperatura massima di archiviazione (GiB) 4	Larghezza di banda basale/burst (Gbps) 5	Motori di supporto 6
db.r8g.24xlarge	96	768	-	1484	40	5.0.0 e 8.0.0
db.r8g.48xlarge	192	1536	-	2967	50	5.0.0 e 8.0.0
R6G — Classe di istanza di generazione attuale basata su Graviton2 Memory-Optimized						
db.r6g.large	2	16	-	32	0,75/10	4.0.0, 5.0.0 e 8.0.0
db.r6g.xlarge	4	32	-	63	1,25/10	4.0.0, 5.0.0 e 8.0.0
db.r6g.2xlarge	8	64	-	126	2,5/10	4.0.0, 5.0.0 e 8.0.0
db.r6g.4xlarge	16	128	-	252	5.0/10	4.0.0, 5.0.0 e 8.0.0
db.r6g.8xlarge	32	256	-	504	12	4.0.0, 5.0.0 e 8.0.0
db.r6g.12xlarge	48	384	-	756	20	4.0.0, 5.0.0 e 8.0.0
db.r6g.16xlarge	64	512	-	1008	25	4.0.0, 5.0.0 e 8.0.0
R6GD — Classe di istanza di generazione attuale basata su Graviton2 NVMe-backed						

Classe di istanza	vCPU[1]	Memoria (GiB) 2	Cache SSD NVMe a più livelli (GiB) 3	Temperatura massima di archiviazione (GiB) 4	Larghezza di banda basale/burst (Gbps) 5	Motori di supporto 6
db.r6gd.xlarge	4	32	173	64	1.25/10	5.0.0 e 8.0.0
db.r6gd.2xlarge	8	64	346	128	2,5/10	5.0.0 e 8.0.0
db.r6gd.4xlarge	16	128	694	256	5,0/10	5.0.0 e 8.0.0
db.r6gd.8xlarge	32	256	1388	512	12	5.0.0 e 8.0.0
db.r6gd.12xlarge	48	384	2082	768	20	5.0.0 e 8.0.0
db.r6gd.16xlarge	64	512	2776	1.024	25	5.0.0 e 8.0.0
R5 — Classe di istanza di generazione precedente Memory-Optimized						
db.r5.large	2	16	-	31	0,75/10	3.6.0, 4.0.0, 5.0.0 e 8.0.0
db.r5.xlarge	4	32	-	62	1,25/10	3.6.0, 4.0.0, 5.0.0 e 8.0.0
db.r5.2xlarge	8	64	-	124	2,5/10	3.6.0, 4.0.0, 5.0.0 e 8.0.0

Classe di istanza	vCPU[1]	Memoria (GiB) 2	Cache SSD NVMe a più livelli (GiB) 3	Temperatura massima di archiviazione (GiB) 4	Larghezza di banda basale/burst (Gbps) 5	Motori di supporto 6
db.r5.4xlarge	16	128	-	249	5,0/10	3.6.0, 4.0.0, 5.0.0 e 8.0.0
db.r5.8xlarge	32	256	-	504	10	3.6.0, 4.0.0, 5.0.0 e 8.0.0
db.r5.12xlarge	48	384	-	748	12	3.6.0, 4.0.0, 5.0.0 e 8.0.0
db.r5.16xlarge	64	512	-	1008	20	3.6.0, 4.0.0, 5.0.0 e 8.0.0
db.r5.24xlarge	96	768	-	1500	25	3.6.0, 4.0.0, 5.0.0 e 8.0.0
R4 — Classe di Memory-Optimized istanza di generazione precedente Le istanze R4 sono supportate solo nella versione del motore 3.6. Tieni presente che Amazon DocumentDB 3.6 termina il supporto standard il 30 marzo 2026. Il supporto esteso (a pagamento) è disponibile fino a marzo 2029.						
db.r4.large	2	15,25	-	30	0,75/10	solo 3.6.0
db.r4.xlarge	4	30,5	-	60	1,25/10	Solo 3.6.0

Classe di istanza	vCPU[1]	Memoria (GiB) 2	Cache SSD NVMe a più livelli (GiB) 3	Temperatura massima di archiviazione (GiB) 4	Larghezza di banda basale/burst (Gbps) 5	Motori di supporto 6
db.r4.2xlarge	8	61	-	120	2,5/10	Solo 3.6.0
db.r4.4xlarge	16	122	-	240	5,0 /10	Solo 3.6.0
db.r4.8xlarge	32	244	-	480	10	Solo 3.6.0
db.r4.16xlarge	64	488	-	960	25	Solo 3.6.0
T4G — Classi Burstable Performance Instance di ultima generazione basate su Graviton2 T-series le istanze vengono eseguite in modalità Unlimited CPU burst. L'utilizzo in modalità burst oltre la linea di base viene fatturato a parte. Non supportato per Global Clusters.						
db.t4g.medium	2	4	-	8.13	0,256/5	4.0.0, 5.0.0 e 8.0.0
T3 — Classi di istanze Burstable Performance di generazione precedente						
db.t3.medium	2	4	-	7.5	0,256/5	3.6.0, 4.0.0, 5.0.0 e 8.0.0

Classe di istanza	vCPU[1]	Memoria (GiB) 2	Cache SSD NVMe a più livelli (GiB) 3	Temperatura massima di archiviazione (GiB) 4	Larghezza di banda basele/burst (Gbps) 5	Motori di supporto 6

1. vCPU: il numero di unità di elaborazione centrale (CPU) virtuali. Una CPU virtuale è un'unità di capacità che puoi utilizzare per confrontare le classi di istanza. Invece di acquistare o affittare un determinato processore da utilizzare per vari mesi o anni, si affitta la capacità su base oraria. Il nostro obiettivo è fornire una quantità coerente di capacità di CPU, indipendentemente dall'hardware reale alla base.
2. Memoria (GiB): la RAM, in gigabyte, allocata all'istanza. Spesso c'è un rapporto costante tra memoria e vCPU.
3. Cache SSD NVMe a più livelli: spazio sul volume SSD, misurato in gigabyte, allocato come cache estesa per l'archiviazione di dati temporanei. Questa cache è NVMe-backed disponibile solo in alcune istanze.
4. Temperatura massima di archiviazione (GiB): lo spazio, misurato in gigabyte, allocato all'istanza per l'archiviazione temporanea di file non persistenti. Ad NVMe-backed esempio, questo storage è ospitato su un volume SSD. NVMe-based In tutti gli altri casi, è ospitato su Amazon Elastic Block Store (EBS).
5. Larghezza di banda baseline/burst (Gbps): la larghezza di banda burst rappresenta la larghezza di banda massima in gigabit al secondo. Dividi per 8 per ottenere il throughput previsto in gigabyte al secondo. Le istanze di dimensioni pari o inferiori a 4xlarge hanno una larghezza di banda di base. Per soddisfare una domanda aggiuntiva, possono utilizzare un meccanismo di I/O credito di rete per superare la larghezza di banda di base. Le istanze possono utilizzare la larghezza di banda burst per un periodo di tempo limitato, in genere da 5 a 60 minuti, a seconda delle dimensioni dell'istanza.
6. Motori di supporto: i motori Amazon DocumentDB che supportano la classe di istanze.

NVMe-backed istanze

NVMe-backed le istanze offrono prestazioni di interrogazione fino a 7 volte più veloci per le applicazioni con set di dati di grandi dimensioni che superano la memoria di un'istanza normale. Queste istanze sfruttano lo storage SSD (Solid-State Drive) locale non volatile memory express

(NVMe) disponibile sulle istanze r6gd per archiviare dati temporanei, ridurre l'accesso allo storage basato sulla rete e migliorare la latenza di lettura e il throughput.

Lo spazio SSD locale è diviso in due sezioni:

- Cache su più livelli: circa il 73% dell'SSD locale è allocato come cache del database, consentendo al sistema di archiviare fino a cinque volte più pagine del database rispetto alla sola memoria principale. L'SSD locale funge da cache di secondo livello, mentre la buffer cache in memoria esistente rimane la cache di primo livello. La query accede all'archiviazione esterna solo in caso di errore sia nella cache buffer che nella cache SSD.
- Archiviazione temporanea: il restante 27% è riservato all'archiviazione temporanea di file non persistente, utilizzata per query complesse che comportano ordinamenti o operazioni ad alta intensità di risorse come la creazione di indici. In casi normali, lo spazio temporaneo risiede su un volume Amazon Elastic Block Store (EBS). L'archiviazione temporanea ospitata localmente sull'SSD riduce la latenza delle query che coinvolgono gli ordinamenti fino a due volte e accelera le operazioni che richiedono molte risorse, come la creazione di indici.

I dettagli relativi al tipo di NVMe-backed istanze supportate e alle relative specifiche sono disponibili in. Specifiche della classe di istanza

Argomenti

- Casi d'uso consigliati per le istanze NVMe-backed
- Utilizzo delle istanze con Amazon DocumentDB NVMe-backed
- Monitoraggio delle istanze NVMe-backed

Casi d'uso consigliati per le istanze NVMe-backed

Usa NVMe-backed le istanze nei seguenti scenari:

- Read-heavy carichi di lavoro: se il carico di lavoro è ad alta intensità di lettura e il set di dati è più grande della cache del buffer, come indicato da `ReadIOPS` metriche basse `BufferCacheHitRatio` e alte, le istanze possono offrire vantaggi in termini di prestazioni. NVMe-backed
- Update-heavy carichi di lavoro: se il carico di lavoro richiede un elevato numero di aggiornamenti e la Garbage Collection non è in grado di tenere il passo a causa della latenza di lettura sullo storage di rete, le istanze potrebbero contribuire a mitigare il problema. NVMe-backed

NVMe-backed le istanze possono beneficiare di vari casi d'uso, tra cui:

- Internet-scale applicazioni: applicazioni come l'elaborazione dei pagamenti, la fatturazione e l'e-commerce con contratti di servizio (SLA) a prestazioni rigorose possono sfruttare i vantaggi prestazionali delle istanze. NVMe-backed
- Real-time dashboard di reporting: le dashboard che eseguono centinaia di query per la metrics/data raccolta possono trarre vantaggio dalla bassa latenza e dall'elevato throughput delle istanze. NVMe-backed
- Applicazioni di intelligenza artificiale generativa: le applicazioni che utilizzano la ricerca vettoriale per trovare i vicini esatti o più vicini su milioni di incorporamenti vettoriali possono sfruttare le elevate prestazioni delle istanze. NVMe-backed

Utilizzo delle istanze con Amazon DocumentDB NVMe-backed

Per utilizzare le NVMe-backed istanze di Amazon DocumentDB:

- Crea un cluster Amazon DocumentDB e aggiungi una delle classi di istanze. NVMe-backed Per ulteriori informazioni, consulta Creazione di un cluster Amazon DocumentDB.
- In alternativa, modifica un cluster Amazon DocumentDB esistente per utilizzare una delle classi di NVMe-backed istanza. Per ulteriori informazioni, consulta Modifica di un cluster Amazon DocumentDB.

Per verificare la disponibilità delle NVMe-backed istanze in diverse AWS regioni, consulta. Classi di istanze supportate per regione

Se desideri tornare da un' NVMe-backed istanza a un'istanza normale, modifica la classe di istanza di database della tua istanza Amazon DocumentDB con una classe di istanza simile senza lo storage NVMe. Ad esempio, se la tua classe di istanza corrente è 'db.r6gd.4xlarge', scegli 'db.r6g.4xlarge' per tornare indietro. Per ulteriori informazioni, consulta Modifica di un cluster Amazon DocumentDB.

Monitoraggio delle istanze NVMe-backed

Oltre alle normali metriche delle istanze disponibili in Amazon CloudWatch, NVMe-backed le istanze emettono parametri aggiuntivi specifici per lo storage NVMe-based SSD, gli IOPS e la velocità effettiva.

```
NVMeStorageCacheHitRatio
```

```
FreeNVMeStorage
ReadIOPSNVMeStorage
ReadLatencyNVMeStorage
ReadThroughputNVMeStorage
WriteIOPSNVMeStorage
WriteLatencyNVMeStorage
WriteThroughputNVMeStorage
```

Per ulteriori informazioni su queste metriche, consulta NVMe-backed metriche dell'istanza

Gestione dei gruppi di sottoreti Amazon DocumentDB

Un cloud privato virtuale (VPC) è una rete virtuale dedicata al tuo. Account AWSÈ logicamente isolato dalle altre reti virtuali nel AWS cloud. Puoi avviare AWS le tue risorse, come i cluster Amazon DocumentDB, nel tuo Amazon VPC. Puoi specificare un intervallo di indirizzi IP per il VPC, aggiungere sottoreti, associare gruppi di sicurezza e configurare tabelle di routing.

Una sottorete è un intervallo di indirizzi IP nel tuo Amazon VPC. Puoi avviare AWS risorse in una sottorete specificata. Utilizza una sottorete pubblica per le risorse che devono essere connesse a Internet. Utilizza una sottorete privata per le risorse che non saranno connesse a Internet. Per ulteriori informazioni sulle sottoreti pubbliche e private, consulta VPC and Subnet Basics nella Amazon Virtual Private Cloud User Guide.

Un gruppo di sottoreti del database è una raccolta di sottoreti creata in un VPC, che è possibile indicare per i cluster. Un gruppo di sottoreti consente di specificare un determinato VPC quando si creano i cluster. Se usi il gruppo di sottoreti `default`, esso include tutte le sottoreti nel VPC.

Ogni gruppo di sottoreti database dovrebbe avere almeno due zone di disponibilità in una data regione . Quando si crea un cluster DB in un VPC, è necessario selezionare un gruppo di sottoreti DB. Amazon DocumentDB utilizza quel gruppo di sottorete DB e la tua zona di disponibilità preferita per selezionare una sottorete e un indirizzo IP all'interno di quella sottorete da associare al cluster. Se l'istanza primaria fallisce, Amazon DocumentDB può promuovere un'istanza di replica corrispondente come nuova primaria. È possibile creare una nuova istanza di replica usando un indirizzo IP della sottorete in cui si trovava l'istanza primaria precedente.

Quando Amazon DocumentDB crea un'istanza in un VPC, assegna un'interfaccia di rete al cluster utilizzando un indirizzo IP selezionato dal gruppo di sottorete del database. Usa sempre il nome DNS perché l'indirizzo IP sottostante può cambiare durante il failover. Per ulteriori informazioni, consulta Endpoint Amazon DocumentDB.

Per informazioni sulla creazione di VPC e sottoreti personalizzati, consulta Working with VPC and Subnet nella Amazon Virtual Private Cloud User Guide.

Argomenti

- Creazione di un gruppo di sottoreti Amazon DocumentDB
- Descrizione di un gruppo di sottoreti Amazon DocumentDB
- Modifica di un gruppo di sottoreti Amazon DocumentDB
- Eliminazione di un gruppo di sottoreti Amazon DocumentDB

Creazione di un gruppo di sottoreti Amazon DocumentDB

Quando crei un cluster Amazon DocumentDB, devi scegliere un Amazon VPC e il corrispondente gruppo di sottorete all'interno di tale Amazon VPC per avviare il cluster. Le sottoreti determinano la zona di disponibilità e l'intervallo IP all'interno della zona di disponibilità che desideri utilizzare per avviare un'istanza.

Un gruppo di sottoreti è un insieme denominato di sottoreti (o AZ) che consente di specificare le zone di disponibilità da utilizzare per avviare le istanze Amazon DocumentDB. Ad esempio, in un cluster con tre istanze, si consiglia di eseguire il provisioning di ciascuna di queste istanze in AZ separati, in modo da ottimizzare l'elevata disponibilità. Pertanto, se una singola AZ fallisce, influirà solo su una singola istanza.

Le istanze Amazon DocumentDB possono essere eseguite in un massimo di tre zone di disponibilità. Anche se un gruppo di sottoreti ha più di tre sottoreti, puoi utilizzarne solo tre per creare un cluster Amazon DocumentDB. Quando crei un gruppo di sottoreti, scegli solo le tre sottoreti in cui desideri distribuire le istanze.

Ad esempio: viene creato un cluster e Amazon DocumentDB sceglie aS {1A, 1B e 1C}. Se tenti di creare un'istanza in AZ {1D} la chiamata API non riuscirà. Tuttavia, se scegli di creare un'istanza, senza specificare la particolare AZ, Amazon DocumentDB sceglierà un'AZ per tuo conto. Amazon DocumentDB utilizza un algoritmo per bilanciare il carico delle istanze su AZ per aiutarti a raggiungere un'elevata disponibilità. Se viene eseguito il provisioning di tre istanze, per impostazione predefinita, il provisioning verrà eseguito su tre AZ e non verrà eseguito il provisioning tutte in una singola AZ.

Best practice

- A meno che non si abbia un motivo specifico, creare sempre un gruppo di sottoreti con tre sottoreti. Ciò garantisce che i cluster con tre o più istanze siano in grado di raggiungere una maggiore disponibilità poiché le istanze verranno eseguite su tre AZ.
- Distribuisci sempre le istanze su più AZ per ottenere una disponibilità più elevata. Non posizionare mai tutte le istanze di un cluster in una singola AZ.
- Poiché gli eventi di failover possono verificarsi in qualsiasi momento, non devi presumere che un'istanza primaria o le istanze di replica siano sempre in una determinata AZ.

Come creare un gruppo di sottorete

Puoi utilizzare Console di gestione AWS o AWS CLI per creare un gruppo di sottoreti Amazon DocumentDB:

Using the Console di gestione AWS

Utilizza i passaggi seguenti per creare un gruppo di sottoreti Amazon DocumentDB.

Come creare un gruppo di sottoreti di Amazon DocumentDB

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Dal riquadro di navigazione, scegliere Subnet group (Gruppo di sottoreti), quindi scegliere Create (Crea).

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Dalla pagina Create subnet group (Crea gruppo di sottorete):

   a. Nella sezione Subnet group details (Dettagli del gruppo di sottoreti):

      i. Nome: inserisci un nome significativo per il gruppo di sottorete.

      ii. Descrizione: immettere una descrizione del gruppo di sottoreti.

   b. Nella sezione Add subnets (Aggiungi sottoreti):

i. VPC: nell'elenco, scegli un VPC per questo gruppo di sottorete.

ii. Esegui una delle seguenti operazioni:

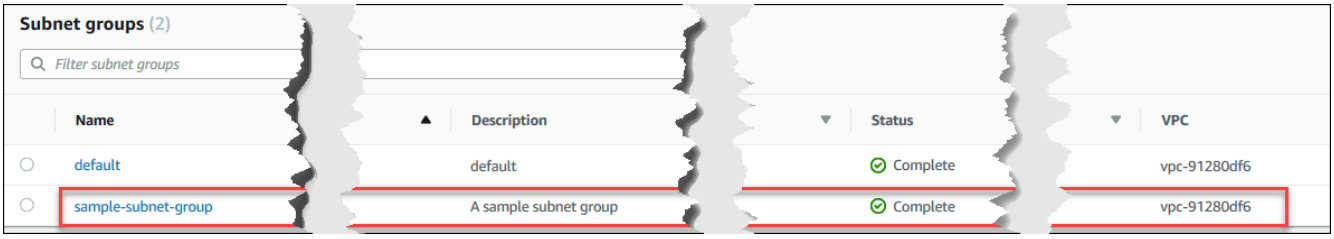
- Per includere tutte le sottoreti nel VPC selezionato, scegliere Add all the subnets related to this VPC (Aggiungi tutte le sottoreti correlate a questo VPC).
- Per specificare le sottoreti per questo gruppo di sottoreti, eseguire la procedura seguente per ogni zona di disponibilità in cui si desidera includere le sottoreti. È necessario includere almeno due zone di disponibilità.

  A. Zona di disponibilità: nell'elenco, scegli una zona di disponibilità.

  B. Subnet: nell'elenco, scegli una sottorete dalla zona di disponibilità scelta per questo gruppo di sottoreti.

  C. Scegliere Add subnet (Aggiungi sottorete).

4. Scegli Create (Crea). Quando viene creato, il gruppo di sottoreti viene elencato con gli altri gruppi di sottoreti.



Using the AWS CLI

Prima di poter creare un gruppo di sottoreti utilizzando il AWS CLI, è necessario innanzitutto determinare quali sottoreti sono disponibili. Eseguire la seguente AWS CLI operazione per elencare le zone di disponibilità e le relative sottoreti.

Parametri:

- **`--db-subnet-group`**—Facoltativo. L'indicazione di un determinato gruppo di sottoreti elencherà le zone di disponibilità e le sottoreti per quel gruppo. L'omissione di questo parametro elencherà zone di disponibilità e le sottoreti per tutti i gruppi di sottoreti. L'indicazione del gruppo di sottoreti `default` elenca tutte le sottoreti del VPC.

Example

Per Linux, macOS o Unix:

```
aws docdb describe-db-subnet-groups \
    --db-subnet-group-name default \
    --query 'DBSubnetGroups[*].[DBSubnetGroupName,Subnets[*].
[SubnetAvailabilityZone.Name,SubnetIdentifier]]'
```

Per Windows:

```
aws docdb describe-db-subnet-groups ^
    --db-subnet-group-name default ^
    --query 'DBSubnetGroups[*].[DBSubnetGroupName,Subnets[*].
[SubnetAvailabilityZone.Name,SubnetIdentifier]]'
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
[
    [
        "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"
            ]
```

```
        ]
    ]
]
```

Utilizzando l'output dall'operazione precedente, è possibile creare un nuovo gruppo di sottoreti. Il nuovo gruppo di sottoreti deve includere sottoreti da almeno due zone di disponibilità.

Parametri:

- **`--db-subnet-group-name`**: obbligatorio. Il nome di questo gruppo di sottoreti.
- **`--db-subnet-group-description`**: obbligatorio. La descrizione di questo gruppo di sottoreti.
- **`--subnet-ids`**: obbligatorio. Un elenco di sottoreti da includere in questo gruppo di sottoreti. Esempio: `subnet-53ab3636`.
- --Tag: opzionale. Un elenco di tag (coppie chiave/valore) da collegare a questo gruppo di sottoreti.

Il codice seguente crea il gruppo di sottoreti `sample-subnet-group` con tre sottoreti `subnet-4e26d263`, `subnet-afc329f4` e `subnet-b3806e8f`.

Per Linux, macOS o 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
```

Per 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
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
```

```
    "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"
    }
}
```

Descrizione di un gruppo di sottoreti Amazon DocumentDB

Puoi utilizzare Console di gestione AWS o the AWS CLI per ottenere i dettagli di un gruppo di sottoreti Amazon DocumentDB.

Using the Console di gestione AWS

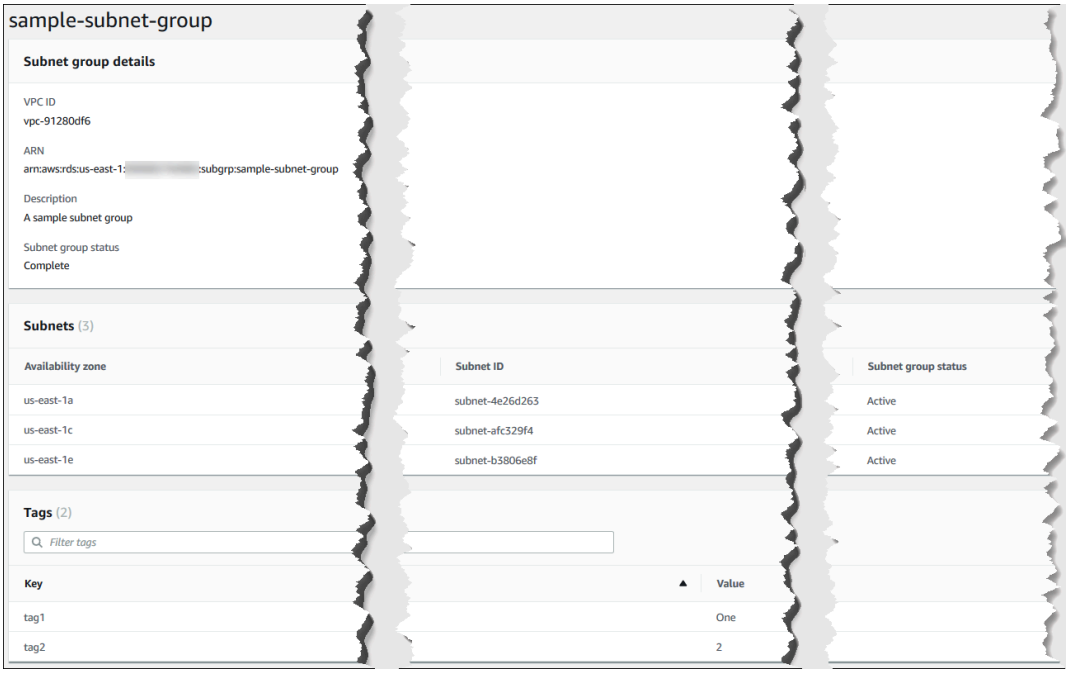
La procedura seguente mostra come ottenere i dettagli di un gruppo di sottoreti Amazon DocumentDB.

Per individuare i dettagli di un gruppo di sottoreti

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).

> **Tip**
>
> Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Per vedere i dettagli di un gruppo di sottoreti, selezionare il nome di tale gruppo di sottoreti.



Using the AWS CLI

Per trovare i dettagli di un gruppo di sottoreti Amazon DocumentDB, utilizza l'`describe-db-subnet-groups`operazione con il seguente parametro.

Parametro

- `--db-subnet=group-name`—Facoltativo. Se incluso, sono elencati i dettagli per il gruppo di sottoreti designato. Se omesso, sono elencati i dettagli di un massimo di 100 gruppi di sottoreti.

Example

Il codice seguente elenca i dettagli per il gruppo di sottoreti `sample-subnet-group` che abbiamo creato nella sezione Creazione di un gruppo di sottoreti Amazon DocumentDB.

Per Linux, macOS o Unix:

```
aws docdb describe-db-subnet-groups \
    --db-subnet-group-name sample-subnet-group
```

Per Windows:

```
aws docdb describe-db-subnet-groups ^
    --db-subnet-group-name sample-subnet-group
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "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"
    }
}
```

Modifica di un gruppo di sottoreti Amazon DocumentDB

Puoi utilizzare Console di gestione AWS o AWS CLI per modificare la descrizione di un gruppo di sottoreti o per aggiungere o rimuovere sottoreti da un gruppo di sottoreti Amazon DocumentDB. Tuttavia, non è possibile modificare il gruppo di sottoreti `default`.

Using the Console di gestione AWS

Puoi utilizzare il Console di gestione AWS per modificare la descrizione di un gruppo di sottoreti o per aggiungere e rimuovere sottoreti. Ricorda che, al termine di questa operazione è necessario disporre di almeno due zone di disponibilità associate al gruppo di sottoreti.

Per modificare il gruppo di sottoreti

1. Accedi a e apri la Console di gestione AWS console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti). Quindi scegliere il pulsante a sinistra del nome del gruppo di sottoreti. Ricorda che non è possibile modificare il gruppo di sottoreti `default`.

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Scegli Actions (Operazioni), quindi Modify (Modifica).
4. Descrizione: per modificare la descrizione del tuo gruppo di sottorete, inserisci una nuova descrizione.

5. Per modificare le sottoreti associate al gruppo di sottoreti, nella sezione Add subnets (Aggiungi sottoreti) , eseguire una o più delle operazioni seguenti:
   - Per rimuovere tutte le sottoreti da questo gruppo di sottoreti, scegliere Remove all (Rimuovi tutte).
   - Per rimuovere sottoreti specifiche da questo gruppo di sottoreti, scegliere Remove (Rimuovi) per ciascuna sottorete da rimuovere.
   - Per aggiungere tutte le sottoreti associate a questo VPC, scegliere Add all the subnets related to this VPC (Aggiungi tutte le sottoreti correlate a questo VPC).
   - Per aggiungere sottoreti specifiche a questo gruppo di sottoreti, eseguire la procedura seguente per ogni zona di disponibilità a cui si desidera aggiungere una sottorete.
     a. Zona di disponibilità: nell'elenco, scegli una nuova zona di disponibilità.
     b. Subnet: nell'elenco, scegli una sottorete dalla zona di disponibilità scelta per questo gruppo di sottoreti.
     c. Scegliere Add subnet (Aggiungi sottorete).
6. Nella finestra di dialogo di conferma:
   - Per apportare tali modifiche al gruppo di sottoreti, scegliere Modify (Modifica).
   - Per mantenere il gruppo di sottoreti invariato, scegliere Cancel (Annulla).

Using the AWS CLI

È possibile utilizzare il AWS CLI per modificare la descrizione di un gruppo di sottoreti o per aggiungere e rimuovere sottoreti. Ricorda che, al termine di questa operazione è necessario disporre di almeno due zone di disponibilità associate al gruppo di sottoreti. Non è possibile modificare il gruppo di sottoreti `default`.

Parametri:

- `--db-subnet-group-name`: obbligatorio. Il nome del gruppo di sottoreti Amazon DocumentDB che stai modificando.
- `--subnet-ids`: obbligatorio. Un elenco di tutte le sottoreti che vuoi aggiungere al gruppo di sottoreti al completamento di questa modifica.

> **Important**
>
> Tutte le sottoreti attualmente nel gruppo di sottoreti che non vengono incluse in questo elenco vengono rimosse dal gruppo di sottoreti. Se si desidera conservare le sottoreti attualmente nel gruppo di sottoreti, è necessario includerle in questo elenco.

- `--db-subnet-group-description`—Facoltativo. La descrizione del gruppo di sottoreti.

Example

Il codice seguente modifica la descrizione e sostituisce le sottoreti esistenti con le sottoreti `subnet-991cb8d0`, `subnet-53ab3636` e `subnet-29ab1025`.

Per Linux, macOS o 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"
```

Per 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"
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON). Questo è lo stesso gruppo di sottoreti che è stato creato nella sezione Creazione di un gruppo di sottoreti Amazon DocumentDB. Tuttavia, le sottoreti nel gruppo di sottoreti vengono sostituite da quelle elencate nell'operazione `modify-db-subnet-group`.

```
{
    "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"
    }
}
```

Eliminazione di un gruppo di sottoreti Amazon DocumentDB

Puoi utilizzare Console di gestione AWS o per eliminare un gruppo di AWS CLI sottoreti di Amazon DocumentDB. Tuttavia, non è possibile eliminare il gruppo di sottoreti `default`.

Using the Console di gestione AWS

È possibile utilizzare il Console di gestione AWS per eliminare un gruppo di sottorete. Tuttavia, non è possibile eliminare il gruppo di sottoreti `default`.

Come eliminare un gruppo di sottoreti

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb

2. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti). Quindi scegliere il pulsante a sinistra del nome del gruppo di sottoreti. Ricorda che non è possibile eliminare il gruppo di sottoreti `default`.

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Scegli Azioni, quindi Elimina.
4. Nella finestra di dialogo di conferma:
   - Per eliminare il gruppo di sottoreti, scegliere Delete (Elimina).
   - Per mantenere il gruppo di sottoreti, scegliere Cancel (Annulla).

Using the AWS CLI

Per eliminare un gruppo di sottoreti Amazon DocumentDB utilizzando il AWS CLI, utilizza l'`delete-db-subnet-group`operazione con il seguente parametro.

Parametro

- `--db-subnet-group-name`: obbligatorio. Il nome del gruppo di sottoreti Amazon DocumentDB da eliminare. Ricorda che non è possibile eliminare il gruppo di sottoreti `default`.

Example

Il codice seguente consente di eliminare `sample-subnet-group`.

Per Linux, macOS o Unix:

```
aws docdb delete-db-subnet-group \
    --db-subnet-group-name sample-subnet-group
```

Per Windows:

```
aws docdb delete-db-subnet-group ^
    --db-subnet-group-name sample-subnet-group
```

L'operazione non produce alcun output.

Disponibilità e replica elevate di Amazon DocumentDB

È possibile ottenere un'elevata disponibilità e scalabilità della lettura in Amazon DocumentDB (con compatibilità MongoDB) utilizzando istanze di replica. Un singolo cluster Amazon DocumentDB supporta una singola istanza primaria e fino a 15 istanze di replica. Queste istanze possono essere distribuite tra le zone di disponibilità all'interno della regione del cluster. L'istanza primaria accetta il traffico in lettura e in scrittura, mentre le istanze di replica accettano solo richieste di lettura.

Il volume cluster si compone di più copie dei dati per il cluster. Tuttavia, i dati nel volume del cluster sono rappresentati come un unico volume logico per l'istanza primaria e per le repliche di Amazon DocumentDB nel cluster. Le istanze di replica sono consistenti finali. Restituiscono risultati di query con un ritardo di replica minimo, generalmente inferiore a 100 millisecondi dopo che l'istanza primaria ha scritto un aggiornamento. Il ritardo di replica varia in base alla velocità di modifica del database. In altre parole, nei periodi in cui si verificano numerose operazioni di scrittura nel database, potresti riscontrare un aumento del ritardo di replica.

Dimensionamento della lettura

Le repliche di Amazon DocumentDB funzionano bene per il ridimensionamento in lettura perché sono completamente dedicate alle operazioni di lettura sul volume del cluster. Le operazioni di lettura sono gestite dall'istanza primaria. Il volume del cluster è condiviso tra tutte le istanze del cluster. Pertanto, non è necessario replicare e mantenere una copia dei dati per ogni replica di Amazon DocumentDB.

Elevata disponibilità

Quando crei un cluster Amazon DocumentDB, a seconda del numero di zone di disponibilità nel gruppo di sottorete (devono essercene almeno due), Amazon DocumentDB effettua il provisioning delle istanze nelle zone di disponibilità. Quando crei istanze nel cluster, Amazon DocumentDB le distribuisce automaticamente tra le zone di disponibilità in un gruppo di sottorete per bilanciare il cluster. Questa azione impedisce inoltre che tutte le istanze si trovino nella stessa zona di disponibilità.

Esempio

Per illustrare meglio questo punto, supponi di creare un cluster che abbia un gruppo di sottorete con tre zone di disponibilità: AZ1, AZ2 e AZ3.

Una volta creata, la prima istanza all'interno del cluster si trova in una delle zone di disponibilità. In questo esempio, si trova nella zona AZ1. La seconda istanza creata è un'istanza di replica e si trova in una delle altre due zone di disponibilità, ad esempio AZ2. La terza istanza creata è un'istanza di replica e si trova nella zona di disponibilità restante, AZ3. Se crei più istanze, queste sono distribuite tra le zone di disponibilità in modo da ottenere il bilanciamento del cluster.

Se si verifica un errore nell'istanza primaria (AZ1), viene attivato un failover e una delle repliche esistenti viene promossa a primaria. Quando la vecchia istanza primaria viene ripristinata, diventa una replica nella stessa zona di disponibilità in cui è stata assegnata (AZ1). Quando esegui il provisioning di un cluster a tre istanze, Amazon DocumentDB continua a preservare quel cluster a tre istanze. Amazon DocumentDB gestisce automaticamente il rilevamento, il failover e il ripristino degli errori delle istanze senza alcun intervento manuale.

Quando Amazon DocumentDB esegue un failover e ripristina un'istanza, l'istanza ripristinata rimane nella zona di disponibilità in cui era stata originariamente fornita. Tuttavia, il ruolo dell'istanza potrebbe cambiare da primaria a replica. Questa operazione viene eseguita per evitare lo scenario in cui una serie di failover provoca il posizionamento di tutte le istanze nella stessa zona di disponibilità.

Puoi specificare le repliche di Amazon DocumentDB come destinazioni di failover. In altre parole, se l'istanza primaria ha esito negativo, la replica o la replica di Amazon DocumentDB specificata da un livello viene promossa all'istanza primaria. Si verifica una breve interruzione durante la quale le richieste di lettura e scrittura inviate all'istanza primaria falliscono con un'eccezione. Se il tuo cluster Amazon DocumentDB non include alcuna replica di Amazon DocumentDB, quando l'istanza primaria ha esito negativo, viene ricreata. La promozione di una replica di Amazon DocumentDB è molto più veloce rispetto alla ricreazione dell'istanza primaria.

Per una disponibilità elevata, crea una o più repliche Amazon DocumentDB della stessa classe di istanza dell'istanza principale in diverse zone di disponibilità.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Comprendere la tolleranza agli errori del cluster Amazon DocumentDB
- Failover di Amazon DocumentDB
  - Controllo della destinazione del failover

Nelle versioni 5.0 e 8.0 di Amazon DocumentDB, le istanze di replica non si riavviano al riavvio dell'istanza primaria. Continuano a fornire richieste di lettura durante i riavvii dell'istanza primaria, mantenendo una maggiore disponibilità.

Alta disponibilità con cluster globali

Per un'elevata disponibilità su più livelli Regioni AWS, puoi configurare cluster globali di Amazon DocumentDB. Ogni cluster globale si estende su più regioni, consentendo letture globali a bassa latenza e il disaster recovery in caso di interruzioni tra regioni. Amazon DocumentDB gestisce automaticamente la replica di tutti i dati e gli aggiornamenti dalla regione principale a ciascuna delle regioni secondarie.

Aggiunta di repliche di

L'istanza primaria è la prima istanza aggiunta al cluster. Ogni istanza aggiunta dopo la prima è un'istanza di replica. Un cluster può avere fino a 15 istanze di replica oltre a quella primaria.

Quando si crea un cluster utilizzando il Console di gestione AWS, contemporaneamente viene creata automaticamente un'istanza primaria. Per creare una replica nello stesso momento in cui viene creato il cluster e l'istanza primaria, scegli Create replica in different zone (Crea replica in una zona diversa). Per ulteriori informazioni, consulta la fase 4.d in Creazione di un cluster Amazon DocumentDB. Per aggiungere altre repliche a un cluster Amazon DocumentDB, consulta. Aggiungere un'istanza Amazon DocumentDB a un cluster

Quando si utilizza il AWS CLI per creare il cluster, è necessario creare in modo esplicito le istanze primarie e di replica. Per ulteriori informazioni, consulta la sezione «Utilizzo del AWS CLI" nei seguenti argomenti:

- Creazione di un cluster Amazon DocumentDB
- Aggiungere un'istanza Amazon DocumentDB a un cluster

Failover di Amazon DocumentDB

In alcuni casi, come alcuni tipi di manutenzione pianificata o nell'improbabile eventualità di un errore di un nodo primario o di una zona di disponibilità, Amazon DocumentDB (con compatibilità MongoDB) rileva l'errore e sostituisce il nodo primario. Durante un failover, il tempo di inattività di scrittura è ridotto al minimo. Questo perché il ruolo del nodo principaleesegue il failover in una delle repliche di lettura invece di creare ed eseguire il provisioning di un nuovo nodo principale. Questa individuazione

degli errori e promozione delle repliche garantisce la possibilità di ricominciare a scrivere nel nuovo nodo primario non appena la promozione è terminata.

Affinché il failover funzioni, il cluster deve avere almeno due istanze: un'istanza primaria e almeno un'istanza di replica.

> **Note**
>
> Questo argomento si applica solo ai cluster originali basati su istanze Amazon DocumentDB. Non si applica ai cluster elastici o globali.

Controllo della destinazione del failover

Amazon DocumentDB fornisce livelli di failover come mezzo per controllare quale istanza di replica viene promossa a primaria quando si verifica un failover.

Livelli di failover

Ogni istanza di replica è associata a un livello di failover (0-15). Quando si verifica un failover a causa di manutenzione o di un improbabile errore hardware, l'istanza primaria esegue il failover su una replica con la priorità più alta (il livello numerato più basso). Se più repliche hanno lo stesso livello di priorità, la replica primaria esegue il failover sulla replica di quel livello con le dimensioni più vicine a quella primaria precedente.

Impostando il livello di failover per un gruppo di repliche selezionate su `0` (priorità massima), puoi assicurare che un failover promuoverà una delle repliche in tale gruppo. Inoltre, per evitare che delle repliche specifiche vengano promosse a istanza primaria in caso di failover, puoi assegnare un livello di priorità basso (numero alto) a queste repliche. Questa funzione è utile nei casi in cui repliche specifiche siano particolarmente utilizzate da un'applicazione e il failover di una di esse potrebbe influire negativamente su un'applicazione critica.

È consigliabile che le repliche che sono potenziali destinazioni di failover utilizzino la stessa classe di istanza dell'istanza principale. Ciò evita un peggioramento delle prestazioni di scrittura se un failover promuove una replica durante gli aggiornamenti del sistema operativo o altre operazioni di manutenzione.

Puoi impostare il livello di failover di un'istanza quando la crei o puoi modificarlo successivamente. Il failover non viene attivato se imposti il livello di failover di un'istanza modificando l'istanza. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Aggiungere un'istanza Amazon DocumentDB a un cluster
- Modifica di un'istanza Amazon DocumentDB

Quando avvii manualmente un failover, puoi controllare quale istanza di replica viene promossa a primaria in due modi: i livelli di failover descritti in precedenza e il parametro `--target-db-instance-identifier`.

`-- target-db-instance-identifier`

Per il testing, puoi forzare un evento di failover utilizzando l'operazione `failover-db-cluster`. Puoi utilizzare il parametro `--target-db-instance-identifier` per specificare quale replica promuovere a istanza primaria. L'utilizzo del parametro `--target-db-instance-identifier` sostituisce il livello di priorità di failover. Se non specifichi il parametro `--target-db-instance-identifier`, il failover primario si conforma al livello di priorità di failover.

Cosa succede durante un failover

Il failover viene gestito automaticamente da Amazon DocumentDB in modo che le applicazioni possano riprendere le operazioni del database il più rapidamente possibile senza interventi amministrativi.

- Se hai un'istanza di replica Amazon DocumentDB nella stessa zona di disponibilità o in una diversa zona di disponibilità quando si esegue il failover: Amazon DocumentDB inverte il record del nome canonico (CNAME) dell'istanza in modo che punti alla replica integra, che a sua volta viene promossa a diventare la nuova primaria. In genere, l'esecuzione completa dell'intero processo di failover impiega meno di 30 secondi.
- Se non disponi di un'istanza di replica Amazon DocumentDB (ad esempio, un cluster di istanze singole): Amazon DocumentDB tenterà di creare una nuova istanza nella stessa zona di disponibilità dell'istanza originale. Questa sostituzione dell'istanza originale viene eseguita in base al tentativo migliore e potrebbe non avvenire, ad esempio, se si verifica un problema che interessa in qualche modo la zona di disponibilità.

L'applicazione deve provare a ristabilire le connessioni al database in caso di perdita della connessione.

Verifica del Failover

Un failover per un cluster promuove una delle repliche di Amazon DocumentDB (istanze di sola lettura) nel cluster come istanza principale (il cluster writer).

Quando l'istanza primaria fallisce, Amazon DocumentDB esegue automaticamente il failover su una replica Amazon DocumentDB, se ne esiste una. Puoi forzare un failover per simulare un guasto di un'istanza primaria per scopi di testing. Ogni istanza in un cluster ha il proprio indirizzo di endpoint. Pertanto, è necessario eliminare e ristabilire tutte le connessioni esistenti che utilizzano tali indirizzi una volta completato il failover..

Per forzare un failover, utilizza l'operazione `failover-db-cluster` con questi parametri.

- `--db-cluster-identifier`: obbligatorio. Il nome del cluster di cui eseguire il failover.
- `--target-db-instance-identifier`—Facoltativo. Il nome dell'istanza da promuovere a istanza primaria.

Example

L'operazione seguente forza un failover del cluster `sample-cluster`. Non specifica quale istanza creare per la nuova istanza primaria, quindi Amazon DocumentDB sceglie l'istanza in base alla priorità del livello di failover.

Per Linux, macOS o Unix:

```
aws docdb failover-db-cluster \
   --db-cluster-identifier sample-cluster
```

Per Windows:

```
aws docdb failover-db-cluster ^
   --db-cluster-identifier sample-cluster
```

L'operazione seguente forza un failover del cluster `sample-cluster`, specificando che `sample-cluster-instance` deve essere promosso al ruolo primario. Annota il valore `"IsClusterWriter": true` nell'output.

Per Linux, macOS o Unix:

```
aws docdb failover-db-cluster \
```

```
   --db-cluster-identifier sample-cluster \
   --target-db-instance-identifier sample-cluster-instance
```

Per Windows:

```
aws docdb failover-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --target-db-instance-identifier sample-cluster-instance
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "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"
    }
}
```

Ritardo di replica

Il ritardo di replica è in genere pari o inferiore a 50 ms. I motivi più comuni dell'aumento del ritardo di replica sono:

- Una velocità di scrittura elevata sul primario che fa sì che le repliche di lettura rimangano indietro rispetto a quelle primarie.
- Conflitto sulle repliche di lettura tra query di lunga durata (ad esempio, scansioni sequenziali di grandi dimensioni, query di aggregazione) e replica in scrittura in entrata.
- Numero molto elevato di interrogazioni simultanee sulle repliche lette.

Per ridurre al minimo il ritardo nella replica, prova queste tecniche di risoluzione dei problemi:

- Se hai una velocità di scrittura elevata o un elevato utilizzo della CPU, aumenta la scalabilità delle istanze nel cluster.
- Se sono presenti query di lunga durata sulle repliche di lettura e aggiornamenti molto frequenti ai documenti oggetto di interrogazione, valuta la possibilità di modificare le query di lunga durata o di eseguirle sulla primary/write replica per evitare conflitti sulle repliche lette.
- Se è presente un numero molto elevato di query simultanee o un elevato utilizzo della CPU solo sulle repliche di lettura, un'altra opzione è ridimensionare il numero di repliche di lettura per distribuire il carico di lavoro.
- Poiché il ritardo di replica deriva da un elevato throughput di scrittura e da query di lunga durata, risolvi i problemi del ritardo di replica utilizzando la metrica in combinazione con lo slow query logger e le metriche/. `DBClusterReplicaLagMaximum` CloudWatch `WriteThroughput` `WriteIOPS`

Assicurati che tutte le repliche siano dello stesso tipo di istanza in modo che un failover del cluster non causi un peggioramento delle prestazioni.

Se scegli tra lo scaling up e lo scaling out (ad esempio, sei istanze più piccole contro tre istanze più grandi), in genere consigliamo di provare a scalare prima (istanze più grandi) prima di eseguire lo scaling out, poiché otterrai una cache di buffer più grande per istanza DB.

In modo proattivo, dovresti impostare un allarme di ritardo di replica e impostarne la soglia su un valore che ritieni sia il limite superiore del ritardo (o «obsoleto») dei tuoi dati sulle istanze di replica prima che inizino a influire sulla funzionalità dell'applicazione. In generale, consigliamo di superare la soglia di ritardo di replica per diversi punti dati prima di creare un allarme, a causa di carichi di lavoro transitori.

> **Note**
>
> Imposta un altro allarme per i ritardi di replica che superano i 10 secondi. Se superi questa soglia per più punti dati, aumenta la scalabilità delle istanze o riduci il throughput di scrittura sull'istanza principale.

Gestione degli eventi di Amazon DocumentDB

Amazon DocumentDB (con compatibilità con MongoDB) registra gli eventi relativi a cluster, istanze, istantanee, gruppi di sicurezza e gruppi di parametri del cluster. Queste informazioni includono la data e l'ora dell'evento, il nome e il tipo di origine dell'evento e un messaggio associato all'evento.

> **Important**
>
> Per alcune funzionalità di gestione, Amazon DocumentDB utilizza una tecnologia operativa condivisa con Amazon RDS e Amazon Neptune. I limiti delle regioni, limiti regolati a livello di regione, sono condivisi tra Amazon DocumentDB, Amazon RDS e Amazon Neptune. Per ulteriori informazioni, consulta Quote regionali.

Argomenti

- Visualizzazione delle categorie di eventi di Amazon DocumentDB
- Visualizzazione degli eventi di Amazon DocumentDB

Visualizzazione delle categorie di eventi di Amazon DocumentDB

Ogni tipo di risorsa Amazon DocumentDB ha tipi specifici di eventi che possono essere associati ad esso. È possibile utilizzare l' AWS CLI `describe-event-categories`operazione per visualizzare la mappatura tra i tipi di eventi e i tipi di risorse Amazon DocumentDB.

Parameters

- **`--source-type`**—Facoltativo. Utilizzare il parametro `--source-type` per visualizzare le categorie di eventi per un determinato tipo di origine. Di seguito sono elencati i valori consentiti:
  - `db-cluster`
  - `db-instance`
  - `db-parameter-group`
  - `db-security-group`
  - `db-cluster-snapshot`
- **`--filters`**—Facoltativo. Per visualizzare le categorie di eventi solo per Amazon DocumentDB, usa il filtro. `--filter Name=engine,Values=docdb`

Example

Il codice seguente elenca le categorie di eventi associati ai cluster.

Per Linux, macOS o Unix:

```
aws docdb describe-event-categories \
    --filter Name=engine,Values=docdb \
    --source-type db-cluster
```

Per Windows:

```
aws docdb describe-event-categories ^
    --filter Name=engine,Values=docdb ^
    --source-type db-cluster
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "EventCategoriesMapList": [
        {
            "EventCategories": [
                "notification",
                "failure",
                "maintenance",
                "failover"
            ],
            "SourceType": "db-cluster"
        }
    ]
}
```

Il codice seguente elenca le categorie di eventi associate a ciascun tipo di sorgente Amazon DocumentDB.

```
aws docdb describe-event-categories
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "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"
```

```
            ]
        }
    ]
}
```

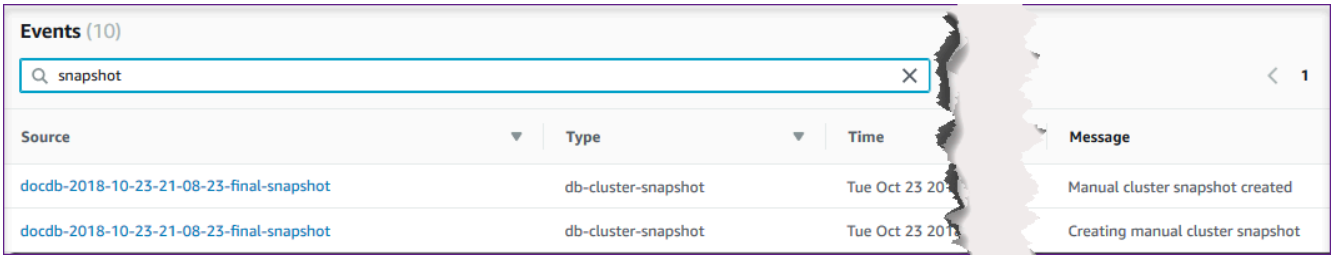
Visualizzazione degli eventi di Amazon DocumentDB

Puoi recuperare gli eventi per le tue risorse Amazon DocumentDB tramite la console Amazon DocumentDB, che mostra gli eventi delle ultime 24 ore. Puoi anche recuperare eventi per le tue risorse Amazon DocumentDB utilizzando il comando AWS CLI describe-events o l'operazione API Amazon DescribeEventsDocumentDB. Se utilizzi l' AWS CLI API di Amazon DocumentDB per visualizzare gli eventi, puoi recuperare gli eventi relativi agli ultimi 14 giorni.

Using the Console di gestione AWS

Per visualizzare tutti gli eventi delle istanze di Amazon DocumentDB nelle ultime 24 ore

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione selezionare Events (Eventi). Gli eventi disponibili sono indicati all'interno di un elenco.
3. Utilizza l'elenco Filter (Filtra) per filtrare gli eventi per tipo. Inserisci un termine nella casella di testo per filtrare ulteriormente i risultati. Ad esempio, la schermata seguente mostra il filtraggio di tutti gli eventi di Amazon DocumentDB per gli eventi snapshot.



Using the AWS CLI

Per visualizzare tutti gli eventi delle istanze di Amazon DocumentDB negli ultimi 7 giorni

Puoi visualizzare tutti gli eventi delle istanze Amazon DocumentDB degli ultimi 7 giorni eseguendo l' AWS CLI operazione describe-events con il `--duration` parametro impostato su (10.080 minuti). `10080`

```
aws docdb describe-events --duration 10080
```

Filtraggio per eventi Amazon DocumentDB

Per visualizzare eventi specifici di Amazon DocumentDB, utilizza l'`describe-events`operazione con i seguenti parametri.

Parameters

- **`--filter`**—Necessario per limitare i valori restituiti agli eventi di Amazon DocumentDB. Utilizzalo **`Name=engine,Values=docdb`** per filtrare tutti gli eventi solo per Amazon DocumentDB.
- **`--source-identifier`**—Facoltativo. L'identificatore dell'origine dell'evento in base a cui vengono restituiti gli eventi. Se omesso, nei risultati sono inclusi gli eventi provenienti da tutte le origini.
- **`--source-type`**—Facoltativo, a meno che non `--source-identifier` sia fornito, quindi obbligatorio. Se viene fornito `--source-identifier`, `--source-type` deve concordare con il tipo di `--source-identifier`. Di seguito sono elencati i valori consentiti:
  - `db-cluster`
  - `db-instance`
  - `db-parameter-group`
  - `db-security-group`
  - `db-cluster-snapshot`

L'esempio seguente elenca tutti gli eventi di Amazon DocumentDB.

```
aws docdb describe-events --filters Name=engine,Values=docdb
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "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"
            ]
        }
    ]
}
```

Per ulteriori informazioni, consulta Verifica degli eventi di Amazon DocumentDB.

Gestione dei gruppi di parametri del cluster Amazon DocumentDB

È possibile gestire la configurazione del motore Amazon DocumentDB utilizzando i parametri in un gruppo di parametri del cluster. Un gruppo di parametri del cluster è una raccolta di valori di configurazione di Amazon DocumentDB che semplificano la gestione dei parametri dei cluster Amazon DocumentDB. I gruppi di parametri del cluster fungono da container per i valori di configurazione del motore applicati a tutte le istanze nel cluster.

In questa sezione viene descritto come creare, visualizzare e modificare i gruppi di parametri del cluster. Viene inoltre illustrato come puoi individuare a quale gruppo di parametri è associato a un determinato cluster.

Argomenti

- Descrizione dei gruppi di parametri del cluster Amazon DocumentDB
- Creazione di gruppi di parametri del cluster Amazon DocumentDB
- Modifica dei gruppi di parametri del cluster Amazon DocumentDB
- Modifica dei cluster Amazon DocumentDB per utilizzare gruppi di parametri di cluster personalizzati
- Copia dei gruppi di parametri del cluster Amazon DocumentDB
- Reimpostazione dei gruppi di parametri del cluster Amazon DocumentDB
- Eliminazione dei gruppi di parametri del cluster Amazon DocumentDB
- Riferimento ai parametri del cluster Amazon DocumentDB

Descrizione dei gruppi di parametri del cluster Amazon DocumentDB

Un gruppo di parametri del `default` cluster viene creato automaticamente quando si crea il primo cluster Amazon DocumentDB in un nuovo motore Regione AWS o si utilizza un nuovo motore. I cluster successivi, creati nella stessa regione e con la stessa versione del motore, vengono creati con il `default` gruppo di parametri del cluster.

Argomenti

- Descrizione dei dettagli di un gruppo di parametri del cluster Amazon DocumentDB
- Determinazione del gruppo di parametri di un cluster Amazon DocumentDB
- Determinazione dei cluster e delle istanze associati a un gruppo di parametri del cluster Amazon DocumentDB

Descrizione dei dettagli di un gruppo di parametri del cluster Amazon DocumentDB

Per descrivere i dettagli di un determinato gruppo di parametri del cluster, completa i passaggi seguenti utilizzando Console di gestione AWS o AWS Command Line Interface (AWS CLI).

Using the Console di gestione AWS

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nel riquadro Parameter groups (Gruppi di parametri), selezionare il gruppo di parametri del cluster di cui si desidera visualizzare i dettagli.
4. La pagina risultante mostra i parametri, l'attività recente e i tag del gruppo di parametri.

   - In Cluster parameters (Parametri cluster) puoi vedere il nome, il valore corrente, i valori consentiti, il tipo di dati e la descrizione del parametro e specifica se il parametro è modificabile. Puoi modificare i singoli parametri selezionando il parametro e scegliendo Edit (Modifica) nella sezione Cluster parameters (Parametri cluster). Per ulteriori informazioni, consulta Modifica dei parametri del cluster Amazon DocumentDB.
   - In Recent events (Eventi recenti), puoi vedere gli eventi più recenti per questo gruppo di parametri. Puoi filtrare questi eventi utilizzando la barra di ricerca in questa sezione. Per ulteriori informazioni, consulta Gestione degli eventi di Amazon DocumentDB.
   - In Tags (Tag) puoi visualizzare i tag di questo gruppo di parametri del cluster. Puoi aggiungere o rimuovere i tag scegliendo Edit (Modifica) nella sezione Tags (Tag). Per ulteriori informazioni, consulta Etichettatura delle risorse di Amazon DocumentDB.

Using the AWS CLI

Puoi utilizzare il `describe-db-cluster-parameter-groups` AWS CLI comando per visualizzare Amazon Resource Name (ARN), la famiglia, la descrizione e il nome di un singolo gruppo di parametri del cluster o di tutti i gruppi di parametri del cluster che hai per Amazon DocumentDB. Puoi anche utilizzare il `describe-db-cluster-parameters` AWS CLI comando

per visualizzare i parametri e i relativi dettagli all'interno di un singolo gruppo di parametri del cluster.

- **`--describe-db-cluster-parameter-groups`**— Per visualizzare un elenco di tutti i gruppi di parametri del cluster e i relativi dettagli.
  - **`--db-cluster-parameter-group-name`**— Facoltativo. Il nome del gruppo di parametri del cluster da descrivere. Se questo parametro viene omesso, vengono descritti tutti i gruppi di parametri del cluster.
- **`--describe-db-cluster-parameters`**— Per elencare tutti i parametri all'interno di un gruppo di parametri e i relativi valori.
  - **`--db-cluster-parameter-group name`**: obbligatorio. Il nome del gruppo di parametri del cluster da descrivere.

Example

Il codice seguente elenca fino a 100 gruppi di parametri del cluster con relativi ARN, famiglia, descrizione e nome.

```
aws docdb describe-db-cluster-parameter-groups
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
          "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

Il codice seguente elenca l'ARN, la famiglia, la descrizione e il nome per `sample-parameter-group`.

Per Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameter-groups \
          --db-cluster-parameter-group-name sample-parameter-group
```

Per Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
          --db-cluster-parameter-group-name sample-parameter-group
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
          "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

Il codice seguente elenca i valori dei parametri in `sample-parameter-group`.

Per Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

Per Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

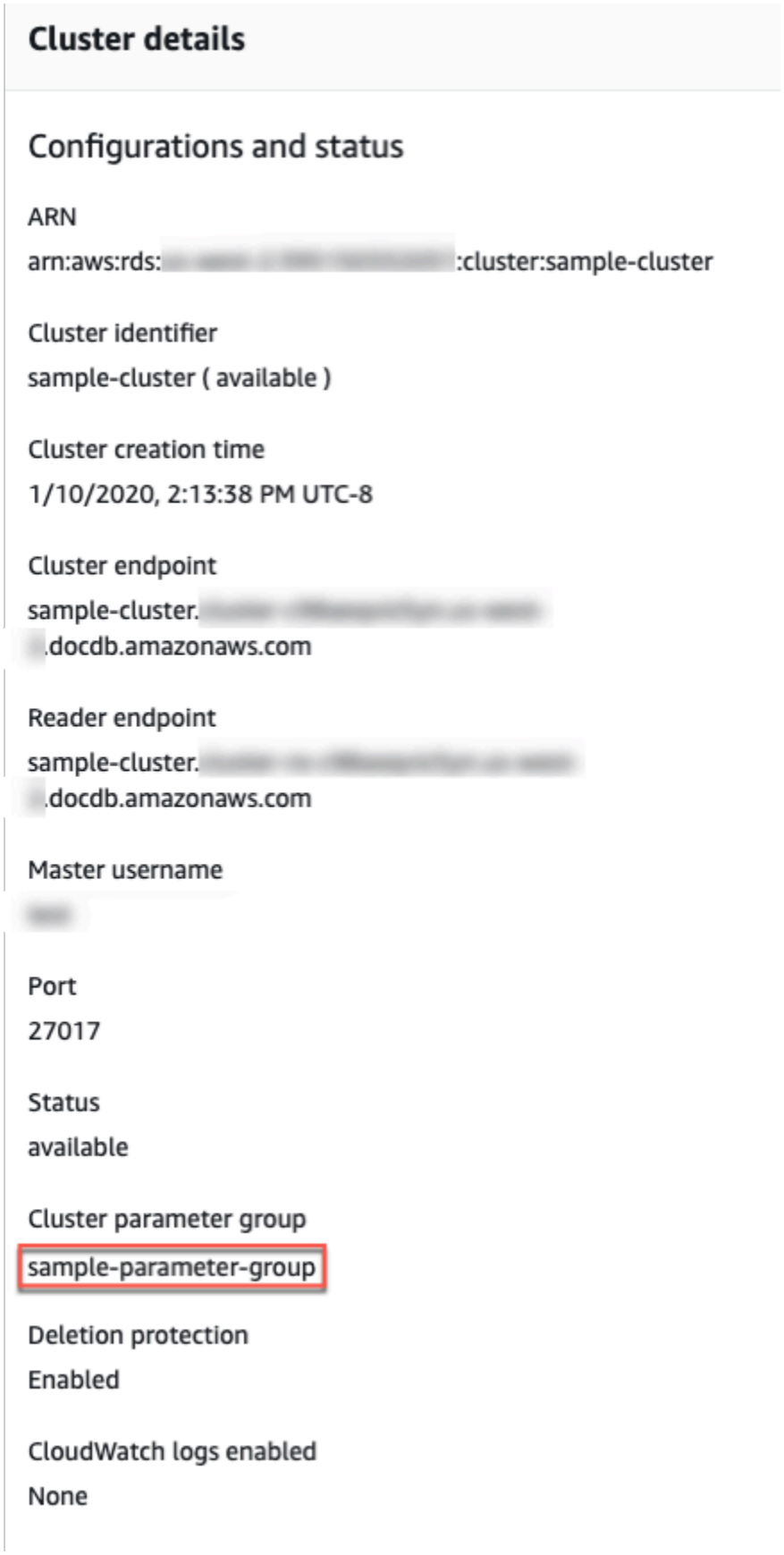
```
{
   "Parameters": [
         {
            "ParameterName": "audit_logs",
            "ParameterValue": "disabled",
            "Description": "Enables auditing on cluster.",
            "Source": "system",
            "ApplyType": "dynamic",
            "DataType": "string",
            "AllowedValues": "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"
         }
   ]
}
```

Determinazione del gruppo di parametri di un cluster Amazon DocumentDB

Per determinare quale gruppo di parametri è associato a un determinato cluster, completare i passaggi seguenti utilizzando Console di gestione AWS o il AWS CLI.

Using the Console di gestione AWS

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Dall'elenco dei cluster seleziona il nome del cluster desiderato.
4. La pagina risultante mostra i dettagli del cluster selezionato. Scorri verso il basso fino a Cluster details (Dettagli del cluster). Nella parte inferiore della sezione, individua il nome del gruppo di parametri sotto Cluster parameter group (Gruppo di parametri del cluster).


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

Il AWS CLI codice seguente determina quale gruppo di parametri governa il cluster`sample-cluster`.

```
aws docdb describe-db-clusters \
      --db-cluster-identifier sample-cluster \
      --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
[
       [
           "sample-cluster",
           "sample-parameter-group"
       ]
]
```

Determinazione dei cluster e delle istanze associati a un gruppo di parametri del cluster Amazon DocumentDB

Per determinare quanti cluster e istanze sono associati a ciascun gruppo di parametri, completa i passaggi seguenti utilizzando il. AWS CLI

Using the AWS CLI

Il AWS CLI codice seguente determina il numero di cluster e istanze associati a. `sample-parameter-group`

Per Linux, macOS o 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
      }) | .[]'
```

Per 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
      }) | .[]'
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
   "parameter_group_name": "sample-parameter-group",
   "total_clusters": 50,
   "total_instances": 150
}
```

Creazione di gruppi di parametri del cluster Amazon DocumentDB

I gruppi di parametri del cluster predefiniti come `default.docdb5.0default.docdb4.0`, o`default.docdb3.6`, vengono creati quando si crea un cluster con una nuova versione del motore e in una nuova. Regione AWS I cluster successivi creati nella stessa regione e con la stessa versione del motore ereditano il gruppo di parametri del `default` cluster. Una volta creati, i gruppi di `default` parametri non possono essere eliminati o rinominati. Puoi modificare il comportamento del motore delle istanze del cluster creando un gruppo di parametri personalizzato con valori di parametro preferiti e collegandolo al tuo cluster Amazon DocumentDB.

La seguente procedura ti guida nella creazione di un gruppo di parametri del cluster personalizzato. Quindi puoi modificare i parametri all'interno del gruppo.

> **Note**
>
> Dopo aver creato un gruppo di parametri del cluster, devi attendere almeno 5 minuti prima di poterlo utilizzare. Ciò consente ad Amazon DocumentDB di completare completamente l'`create`azione prima che il gruppo di parametri del cluster venga utilizzato per un

nuovo cluster. Puoi utilizzare l'`describe-db-cluster-parameter-groups` AWS CLI operazione Console di gestione AWS or the per verificare che il gruppo di parametri del cluster sia stato creato. Per ulteriori informazioni, consulta Descrizione dei gruppi di parametri del cluster Amazon DocumentDB.

Using the Console di gestione AWS

Per creare un gruppo di parametri del cluster

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nel riquadro Cluster parameter groups (Gruppi di parametri del cluster), scegliere Create (Crea).
4. Nel riquadro Create cluster parameter group (Crea gruppo di parametri del cluster) immettere quanto segue:

   a. Nuovo nome del gruppo di parametri del cluster: inserisci un nome per il gruppo di parametri del cluster. Ad esempio, `sample-parameter-group`. I gruppi di parametri del cluster hanno i seguenti vincoli di denominazione:

      - La lunghezza è compresa tra 1 e 255 caratteri alfanumerici.
      - Il primo carattere deve essere una lettera.
      - Non può terminare con un trattino o contenere due trattini consecutivi.

   b. Famiglia: scegli una versione di DocumentDB che intendi utilizzare per il tuo cluster.

   c. Descrizione: fornisci una descrizione per questo gruppo di parametri del cluster.

5. Per creare il gruppo di parametri del cluster, scegli Create (Crea). Per annullare l'operazione, scegli Cancel (Annulla).

6. Dopo aver scelto Create (Crea), nella parte superiore della pagina viene visualizzato il testo seguente per verificare che il gruppo di parametri del cluster sia stato creato:

```
Successfully created cluster parameter group 'sample-parameter-group'.
```

Using the AWS CLI

Per creare un nuovo gruppo di parametri del cluster per i cluster Amazon DocumentDB 4.0, utilizza l' AWS CLI `create-db-cluster-parameter-group`operazione con i seguenti parametri:

- **`--db-cluster-parameter-group-name`**— Il nome del gruppo di parametri del cluster personalizzato. Ad esempio, `sample-parameter-group`.
- **`--db-cluster-parameter-group-family`**— La famiglia di gruppi di parametri del cluster utilizzata come modello per il gruppo di parametri del cluster personalizzato.
- **`--description`**— La descrizione fornita dall'utente per questo gruppo di parametri del cluster. L'esempio seguente utilizza "`Custom docdb4.0 parameter group`».

Per Linux, macOS o 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"
```

Per 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"
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
```

```
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "sample-parameter-group",
        "DBParameterGroupFamily": "docdb5.0",
        "Description": "Custom docdb4.0 parameter group",
        "DBClusterParameterGroupArn": "sample-parameter-group-arn"
    }
}
```

Modifica dei gruppi di parametri del cluster Amazon DocumentDB

Questa sezione spiega come modificare un gruppo di parametri Amazon DocumentDB personalizzato. In Amazon DocumentDB, non è possibile modificare un gruppo di parametri del `default` cluster creato quando si crea per la prima volta un cluster con una nuova versione del motore in una nuova regione. Se il tuo cluster Amazon DocumentDB utilizza il gruppo di parametri del cluster predefinito e desideri modificarne un valore, devi prima creare un nuovo gruppo di parametri o copiare un gruppo di parametri esistente, modificarlo e quindi applicare il gruppo di parametri modificato al cluster.

Completa i passaggi seguenti per modificare un gruppo di parametri del cluster personalizzato. La propagazione delle azioni di modifica potrebbe richiedere del tempo. Attendi che il gruppo di parametri del cluster modificato sia disponibile prima di collegarlo al cluster. È possibile utilizzare l' AWS CLI `describe-db-cluster-parameters`operazione Console di gestione AWS o per verificare che il gruppo di parametri del cluster sia stato modificato. Per ulteriori informazioni, consulta Descrizione dei gruppi di parametri del cluster.

Using the Console di gestione AWS

Segui questi passaggi per modificare un gruppo di parametri Amazon DocumentDB personalizzato. Non puoi modificare un gruppo di parametri `default`. Se desideri modificare un valore nel gruppo di parametri `default`, puoi copiare il gruppo di parametri del cluster predefinito, modificarlo e quindi applicare il gruppo di parametri modificato al cluster. Per ulteriori informazioni sull'applicazione dei gruppi di parametri al cluster, consulta Modifica di un cluster Amazon DocumentDB.

Per modificare un gruppo di parametri del cluster personalizzato

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb

2. Nel riquadro di navigazione sul lato sinistro della console scegli Parameter groups (Gruppi di parametri). Nell'elenco dei gruppi di parametri, scegli il nome del gruppo di parametri da modificare.

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Per ogni parametro nel gruppo di parametri che desideri modificare, procedi nel seguente modo:

   a. Individua il parametro che vuoi modificare e verifica che sia modificabile controllando se è elencato come `true` nella colonna Modifiable (Modificabile).

   b. Se è modificabile, seleziona il parametro e scegli Edit (Modifica) in alto a destra nella pagina della console.

   c. Nella finestra di dialogo Modify **<parameter-name>** (Modifica), apporta le modifiche desiderate. Quindi scegli Modify cluster parameter (Modifica parametro del cluster) oppure scegli Cancel (Annulla) per annullare le modifiche.

Using the AWS CLI

Puoi modificare il`ParameterValue`,`Description`, o `ApplyMethod` di qualsiasi parametro modificabile in un gruppo di parametri del cluster Amazon DocumentDB personalizzato utilizzando. AWS CLI Non puoi apportare modifiche direttamente in un gruppo di parametri del cluster predefinito.

Per modificare i parametri di un gruppo di parametri del cluster personalizzato, utilizza l'operazione `modify-db-cluster-parameter-group` con i parametri seguenti.

- **`--db-cluster-parameter-group-name`**: obbligatorio. Il nome del gruppo di parametri del cluster che stai modificando.
- **`--parameters`**: obbligatorio. I parametri che stai modificando. Per un elenco dei parametri che si applicano a tutte le istanze in un cluster Amazon DocumentDB, consulta. Riferimento ai parametri del cluster Amazon DocumentDB Ogni voce del parametro deve includere:

- **`ParameterName`**— Il nome del parametro che stai modificando.
- **`ParameterValue`**— Il nuovo valore per questo parametro.
- **`ApplyMethod`**— Come si desidera che vengano applicate le modifiche a questo parametro. I valori consentiti sono `immediate` e `pending-reboot`.

> **Note**
>
> I parametri con `ApplyType` per `static` devono avere `ApplyMethod` per `pending-reboot`.

Example- Modifica del valore di un parametro

In questo esempio, puoi elencare i valori dei parametri di `sample-parameter-group` e modificare il parametro `tls`. Quindi, dopo 5 minuti, puoi elencare di nuovo i valori dei parametri di `sample-parameter-group` per visualizzare i valori modificati.

1. Elenca i parametri e i relativi valori di `sample-parameter-group`.

   Per Linux, macOS o Unix:

   ```
   aws docdb describe-db-cluster-parameters \
          --db-cluster-parameter-group-name sample-parameter-group
   ```

   Per Windows:

   ```
   aws docdb describe-db-cluster-parameters ^
          --db-cluster-parameter-group-name sample-parameter-group
   ```

   L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

   ```
   {
          "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. Modifica il parametro `tls` in modo che il valore sia `disabled`.

   Non puoi modificare `ApplyMethod` perché `ApplyType` è `static`.

   Per Linux, macOS o Unix:

```
aws docdb modify-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group \
       --parameters
 "ParameterName"=tls,"ParameterValue"=disabled,"ApplyMethod"=pending-reboot
```

   Per Windows:

```
aws docdb modify-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group ^
       --parameters
 "ParameterName"=tls,"ParameterValue"=disabled,"ApplyMethod"=pending-reboot
```

   L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
       "DBClusterParameterGroupName": "sample-parameter-group"
   }
```

3. Attendi almeno 5 minuti.
4. Elenca i valori dei parametri di `sample-parameter-group` per verificare che il parametro `tls` sia stato modificato.

   Per Linux, macOS o Unix:

   ```
   aws docdb describe-db-cluster-parameters \
          --db-cluster-parameter-group-name sample-parameter-group
   ```

   Per Windows:

   ```
   aws docdb describe-db-cluster-parameters ^
          --db-cluster-parameter-group-name sample-parameter-group
   ```

   L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

   ```
   {
          "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"
              }
          ]
   ```

```
}
```

Modifica dei cluster Amazon DocumentDB per utilizzare gruppi di parametri di cluster personalizzati

Quando crei un cluster Amazon DocumentDB, viene creato automaticamente un gruppo di `default.docdb4.0` parametri per quel cluster. Non è consentito modificare il gruppo di parametri `default`. Puoi invece modificare il tuo cluster Amazon DocumentDB per associarvi un nuovo gruppo di parametri personalizzato.

Questa sezione spiega come modificare un cluster Amazon DocumentDB esistente per utilizzare un gruppo di parametri del cluster personalizzato utilizzando Console di gestione AWS and the AWS Command Line Interface ()AWS CLI.

Using the Console di gestione AWS

Per modificare un cluster Amazon DocumentDB per utilizzare un nuovo gruppo di parametri del cluster non predefinito

1. Prima di iniziare, assicurati di aver creato un cluster Amazon DocumentDB e un gruppo di parametri del cluster. Vedere Creazione di un cluster Amazon DocumentDB e Creazione di gruppi di parametri del cluster Amazon DocumentDB per ulteriori istruzioni.
2. Dopo aver creato il gruppo di parametri del cluster, apri la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb Nel riquadro di navigazione scegliere Clusters (Cluster) per aggiungere il nuovo gruppo di parametri a un cluster.
3. Scegliere il cluster a cui si desidera associare il gruppo di parametri. Scegliere Actions (Operazioni), quindi scegliere Modify (Modifica) per modificare il cluster.
4. In Cluster options (Opzioni cluster), scegliere il nuovo gruppo di parametri a cui si desidera associare il cluster.
5. Scegliere Continue (Continua) per visualizzare un riepilogo delle modifiche.
6. Dopo aver verificato le modifiche, è possibile applicarle immediatamente o durante la successiva finestra di manutenzione in Scheduling of modifications (Pianificazione delle modifiche).
7. Scegliere Modify cluster (Modifica cluster) per aggiornare il cluster con il nuovo gruppo di parametri.

Using the AWS CLI

Prima di iniziare, assicurati di aver creato un cluster Amazon DocumentDB e un gruppo di parametri del cluster. Puoi creare un cluster Amazon DocumentDB utilizzando l' AWS CLI `create-db-cluster`operazione. Puoi creare un gruppo di parametri del cluster utilizzando l' AWS CLI `create-db-cluster-parameter-group`operazione.

Per aggiungere il nuovo gruppo di parametri del cluster al cluster, utilizza l' AWS CLI `modify-db-cluster`operazione con i seguenti parametri.

- --db-cluster-identifier — Il nome del cluster (ad esempio,). `sample-cluster`
- --db-cluster-parameter-group-name — Il nome del gruppo di parametri a cui si desidera associare il cluster (ad esempio,). `sample-parameter-group`

Example

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --db-cluster-parameter-group-name sample-parameter-group
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
"DBCluster": {
    "AvailabilityZones": [
            "us-west-2c",
            "us-west-2b",
            "us-west-2a"
    ],
    "BackupRetentionPeriod": 1,
    "DBClusterIdentifier": "sample-cluster",
    "DBClusterParameterGroup": "sample-parameter-group",
    "DBSubnetGroup": "default",
   ...
}
```

Copia dei gruppi di parametri del cluster Amazon DocumentDB

Puoi creare una copia di un gruppo di parametri del cluster in Amazon DocumentDB utilizzando Console di gestione AWS o AWS Command Line Interface ()AWS CLI.

Using the Console di gestione AWS

La procedura seguente ti guida nella creazione di un nuovo gruppo di parametri del cluster eseguendo una copia di un gruppo di parametri cluster esistente.

Per copiare un gruppo di parametri del cluster

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nel riquadro Cluster parameter groups (Gruppi di parametri del cluster), scegliere il nome del gruppo di parametri del cluster di cui si desidera visualizzare i dettagli.
4. Scegliere Actions (Operazioni), quindi scegliere Copy (Copia) per copiare il gruppo di parametri.
5. In Copy options (Opzioni copia), immettere un nome e una descrizione per il nuovo gruppo di parametri del cluster. Quindi scegliere Copy (Copia) per salvare le modifiche.

Using the AWS CLI

Per copiare un gruppo di parametri del cluster, utilizza l'operazione `copy-db-cluster-parameter-group` con i parametri seguenti.

- **`--source-db-cluster-parameter-group-identifier`**: obbligatorio. Il nome o l'Amazon Resource Name (ARN) del gruppo di parametri del cluster da copiare.

  Se i gruppi di parametri del cluster di origine e di destinazione sono uguali Regione AWS, l'identificatore può essere un nome o un ARN.

  Se i gruppi di parametri del cluster di origine e di destinazione sono diversi Regioni AWS, l'identificatore deve essere un ARN.
- **`--target-db-cluster-parameter-group-identifier`**: obbligatorio. Il nome o l'ARN della copia del gruppo di parametri del cluster.

  Vincoli:
  - Non può essere null o vuoto.
  - Deve contenere da 1 a 255 lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.

- Non può terminare con un trattino o contenere due trattini consecutivi.

- **`--target-db-cluster-parameter-group-description`**: obbligatorio. Una descrizione per la copia del gruppo di parametri del cluster fornita dall'utente.

Example

Il codice seguente crea una copia di `sample-parameter-group`, denominandola `sample-parameter-group-copy`.

Per Linux, macOS o 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"
```

Per 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"
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "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"
    }
}
```

Reimpostazione dei gruppi di parametri del cluster Amazon DocumentDB

Puoi ripristinare alcuni o tutti i valori dei parametri di un gruppo di parametri del cluster Amazon DocumentDB ai valori predefiniti utilizzando Console di gestione AWS o the AWS Command Line Interface (AWS CLI) per reimpostare il gruppo di parametri del cluster.

Using the Console di gestione AWS

Attieniti alla seguente procedura per reimpostare alcuni o tutti i valori di un parametro di un gruppo di parametri del cluster ai valori predefiniti.

Per reimpostare i valori dei parametri di un gruppo di parametri del cluster

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel riquadro di navigazione sul lato sinistro della console scegli Parameter groups (Gruppi di parametri).
3. Nel riquadro Cluster parameter groups (Gruppi di parametri del cluster), scegliere il nome del gruppo di parametri del cluster da reimpostare.
4. Scegliere Actions (Operazioni), quindi scegliere Reset (Reimposta) per reimpostare il gruppo di parametri.
5. Nella pagina di conferma della reimpostazione del gruppo di parametri del cluster restituita, confermare che si desidera reimpostare i valori predefiniti di tutti i parametri del cluster per il gruppo di parametri. Quindi scegliere Reset (Reimposta) per reimpostare il gruppo di parametri. È anche possibile selezionare Cancel (Annulla) per annullare le modifiche.

Using the AWS CLI

Per reimpostare alcuni o tutti i valori di un parametro di un gruppo di parametri del cluster ai valori predefiniti, puoi utilizzare l'operazione `reset-db-cluster-parameter-group` con i parametri seguenti.

- **`--db-cluster-parameter-group-name`**: obbligatorio. Il nome del gruppo di parametri del cluster da reimpostare.
- **`--parameters`**— Facoltativo. Un elenco di `ParameterName` e `ApplyMethod` nel gruppo di parametri del cluster da ripristinare ai valori predefiniti. I parametri statici devono essere impostati su `pending-reboot` per essere applicati al successivo riavvio dell'istanza o quando

viene eseguita la richiestareboot-db-instance. Devi chiamare reboot-db-instance per ogni istanza nel cluster a cui desideri applicare il parametro statico aggiornato.

Questo parametro e --reset-all-parameters sono reciprocamente esclusivi: puoi utilizzarne uno solo, non entrambi.

- **--reset-all-parameters**o **--no-reset-all-parameters** — Facoltativo. Specifica se ripristinare tutti i parametri (--reset-all-parameters) o solo alcuni (--no-reset-all-parameters) ai valori predefiniti. Il parametro --reset-all-parameters e --parameters sono reciprocamente esclusivi: puoi utilizzarne uno solo, non entrambi.

  Quando reimposti l'intero gruppo, i parametri dinamici vengono aggiornati immediatamente. I parametri statici devono essere impostati su pending-reboot per essere applicati al successivo riavvio dell'istanza o quando viene eseguita la richiesta reboot-db-instance. Devi chiamare reboot-db-instance per ogni istanza nel cluster a cui desideri applicare il parametro statico aggiornato.

Example

Esempio 1: reimpostazione di tutti i parametri ai valori predefiniti

Il codice seguente reimposta tutti i parametri nel gruppo di parametri del cluster sample-parameter-group ai valori predefiniti.

Per Linux, macOS o Unix:

```
aws docdb reset-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group \
       --reset-all-parameters
```

Per Windows:

```
aws docdb reset-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group ^
       --reset-all-parameters
```

Esempio 2: reimpostazione di parametri specifici ai valori predefiniti

Il codice seguente reimposta il parametro tls nel gruppo di parametri del cluster sample-parameter-group al valore predefinito.

Per Linux, macOS o 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
```

Per 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
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
       "DBClusterParameterGroupName": "sample-parameter-group"
   }
```

Riavvio di un'istanza cluster

Prima di modificare il valore di un parametro statico, l'istanza cluster deve essere riavviata. Riavvia ogni istanza nel cluster a cui desideri applicare il parametro statico aggiornato.

Per Linux, macOS o Unix:

```
aws docdb reboot-db-instance \
       --db-instance-identifier sample-cluster-instance
```

Per Windows:

```
aws docdb reboot-db-instance ^
       --db-instance-identifier sample-cluster-instance
```

Eliminazione dei gruppi di parametri del cluster Amazon DocumentDB

Puoi eliminare un gruppo di parametri del cluster Amazon DocumentDB personalizzato utilizzando Console di gestione AWS o AWS Command Line Interface ()AWS CLI. Non è possibile eliminare il gruppo di parametri del cluster `default.docdb4.0`.

Using the Console di gestione AWS

Per eliminare un gruppo di parametri del cluster

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nel riquadro Parameter groups (Gruppi di parametri) scegli il pulsante di opzione a sinistra del gruppo di parametri del cluster da eliminare.
4. Scegli Azioni, quindi Elimina.
5. Nel riquadro di conferma Delete (Elimina) scegliere Delete (Elimina) per eliminare il gruppo di parametri del cluster. Per mantenere il gruppo di parametri del cluster, scegli Annulla.

Using the AWS CLI

Per eliminare un gruppo di parametri del cluster, utilizza l'operazione `delete-db-cluster-parameter-group` con il parametro seguente.

- **`--db-cluster-parameter-group-name`**: obbligatorio. Il nome del gruppo di parametri del cluster da eliminare. Deve essere un gruppo di parametri del cluster esistente. Non è possibile eliminare il gruppo di parametri del cluster *`default.docdb4.0`*.

Example- Eliminazione di un gruppo di parametri del cluster

L'esempio seguente descrive le tre fasi che consentono di eliminare un gruppo di parametri del cluster:

1. Ricerca del nome del gruppo di parametri del cluster da eliminare.
2. Eliminazione del gruppo di parametri del cluster specificato.
3. Verifica dell'avvenuta eliminazione del gruppo di parametri del cluster.

1. Cerca il nome del gruppo di parametri del cluster da eliminare.

Il codice seguente elenca i nomi di tutti i gruppi di parametri del cluster.

Per Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameter-groups \
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Per Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

L'output dell'operazione precedente è un elenco dei nomi dei gruppi di parametri del cluster simile al seguente (formato JSON).

```
[
       [
           "default.docdb4.0"
       ],
       [
           "sample-parameter-group"
       ],
       [
           "sample-parameter-group-copy"
       ]
   ]
```

2. Elimina un gruppo di parametri del cluster specifico.

Il codice seguente elimina il gruppo di parametri del cluster `sample-parameter-group-copy`.

Per Linux, macOS o Unix:

```
aws docdb delete-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group-copy
```

Per Windows:

```
aws docdb delete-db-cluster-parameter-group ^
```

```
        --db-cluster-parameter-group-name sample-parameter-group-copy
```

Questa operazione non produce alcun output.

3. Verifica dell'avvenuta eliminazione del gruppo di parametri del cluster specificato.

Il codice seguente elenca i nomi di tutti i gruppi di parametri del cluster rimanenti.

Per Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameter-groups \
        --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Per Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
        --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

L'output dell'operazione precedente è un elenco dei gruppi di parametri del cluster simile al seguente (formato JSON). Il gruppo di parametri del cluster appena eliminato non deve comparire nell'elenco.

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
[
        [
            "default.docdb4.0"
        ],
        [
            "sample-parameter-group"
        ]
   ]
```

Riferimento ai parametri del cluster Amazon DocumentDB

Quando modifichi un parametro dinamico e salvi il gruppo di parametri del cluster, la modifica viene applicata immediatamente, indipendentemente dall'impostazione Apply Immediately (Applica immediatamente). Quando modifichi un parametro statico e salvi il gruppo di parametri del cluster, la modifica del parametro viene applicata dopo che avrai riavviato manualmente l'istanza. Puoi riavviare

un'istanza utilizzando la console Amazon DocumentDB o chiamando esplicitamente. `reboot-db-instance`

La tabella seguente mostra i parametri che si applicano a tutte le istanze in un cluster Amazon DocumentDB.

Parametri a livello di cluster Amazon DocumentDB

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
`audit_logs`	disabled	abilitato, disabilitato, ddl, dml_read, dml_write, tutti, nessuno	Sì	Dinamico	Stringa	Definisce se i log di controllo di Amazon sono abilitati. CloudWatch • **`enabled`**— i log CloudWatch di controllo sono abilitati. • **`disabled`**— i CloudWatch registri di controllo sono disabilitati. • **`ddl`**— il controllo

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
						per gli eventi DDL è abilitato. • **dml_read**— il controllo degli eventi di lettura DML è abilitato. • **dml_write** — il controllo per gli eventi di scrittura DML è abilitato. • **all**— il controllo per tutti gli eventi del database è abilitato. • **none**— il controllo è disabilitato.

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
change_stream_log_retention_duration	10800	3600-604800	Sì	Dinamico	Numero intero	Definisce la durata del tempo (in secondi) in cui il registro del flusso di modifiche viene mantenuto e può essere consumato.

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
`default_collection_compression`	disabled	abilitato, disabilitato (Amazon DocumentDB 5.0)/ zstd, lz4, nessuno (Amazon) DocumentDB8.0	Sì	Dinamico	Stringa	Definisce l'impostazione di compressione predefinita per le nuove raccolte in un cluster • **`enabled`**— La compressione Lz4 è abilitata di default in Amazon DocumentDB 5.0. la compressione zstd è abilitata di default in Amazon DocumentDB 8.0. • **`disabled, none`**— la

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
						compressione è disabilitata per impostazione predefinita (disattivata in Amazon DocumentDB 5.0, nessuna in Amazon DocumentDB 8.0).

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
`profiler`	disabled	abilitato, disabilitato	Sì	Dinamico	Stringa	Abilita il profiling per operazioni lente. • **`enabled`**— le operazioni che richiedono più tempo di un valore di soglia definito dal cliente (ad esempio 100 ms) vengono registrate in Amazon Logs. CloudWatch • **`disabled`**— le operazioni lente non vengono

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
						registrate nei log. CloudWatch
`profiler_sampling_rate`	1.0	0,0 - 1,0	Sì	Dinamico	Float	Definisce la frequenza di campionamento per le operazioni registrate.
`profiler_threshold_ms`	100	50-2147483646	Sì	Dinamico	Numero intero	Definisce la soglia per `profiler`. • Tutte le operazioni superiori a quelle `profiler_threshold_ms` vengono registrate nei log. CloudWatch

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
`planner_version`	3.0	1.0, 2.0, 3.0	Sì	Dinamico	Float	Definisce la versione di Query Planner da utilizzare per le interrogazioni. • **`1.0`**— Supportato in Amazon DocumentDB 5.0. • **`2.0`**— Miglioramenti alla ricerca e all'aggiornamento degli operatori. Supportato in Amazon DocumentDB 5.0. • **`3.0`**— Miglioramenti agli operatori

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
						di ricerca, aggiornamento e aggregazione. Pianificatore di query predefinito in Amazon DocumentDB 8.0.

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
`tls`	abilitato	abilitato, disabilitato, fips-140-3, tls1.2+, tls1.3+	Sì	Statico	Stringa	Definisce se sono necessarie connessioni Transport Layer Security (TLS). • **`enabled`**—Per connettersi sono necessarie connessioni TLS. Per Amazon DocumentDB 5.0 e versioni precedenti, abilita TLS da 1.0 a 1.3, ma utilizza TLS 1.2 o versioni successive. Solo

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
						TLS 1.2 e versioni successive sono supportati su Amazon DocumentDB 5.0 a partire dalla versione secondaria 5.0.1, Amazon DocumentDB 8.0 e successive. • **`disabled`**— Le connessioni TLS non possono essere utilizzate per connettersi. • **`fips-140-3`** —

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
						Per la connessione sono necessarie connessioni TLS con attributi FIPS (Federal Information Processing Standards). Il cluster accetta solo connessioni sicure secondo la pubblicazione FIPS 140-3. È supportato solo a partire dai cluster Amazon

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
						DocumentDB 5.0 (versione del motore 3.0.3727) nelle seguenti regioni: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1. • **tls1.2+**— Per la connessione sono necessarie connessioni TLS che utilizzano TLS versione 1.2 e

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
						successive. È supportato solo a partire da Amazon DocumentDB 4.0 (versione del motore 2.0.10980) e Amazon DocumentDB 5.0 (versione del motore 3.0.11051). • **`tls1.3+`**— Per la connessione sono necessarie connessioni TLS che utilizzano TLS versione

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
						1.3 e successive. È supportato solo a partire da Amazon DocumentDB 4.0 (versione del motore 2.0.10980) e Amazon DocumentDB 5.0 (versione del motore 3.0.11051).

Parametro	Valore predefinito	Valori validi	Modificabile	Applica tipo	Tipo di dati	Description
`ttl_monitor`	abilitato	abilitato, disabilitato	Sì	Dinamico	Stringa	Definisce se il monitoraggio Time to Live (TTL) è abilitato per il cluster. • **`enabled`**— Il monitoraggio TTL è abilitato. • **`disabled`**— Il monitoraggio TTL è disabilitato.

Modifica dei parametri del cluster Amazon DocumentDB

In Amazon DocumentDB, i gruppi di parametri del cluster sono costituiti da parametri che si applicano a tutte le istanze create nel cluster. Per i gruppi di parametri del cluster personalizzati, puoi modificare un valore di parametro in qualsiasi momento o reimpostare tutti i valori dei parametri sulle impostazioni predefinite per i gruppi di parametri creati. Questa sezione descrive come visualizzare i parametri che compongono un gruppo di parametri del cluster Amazon DocumentDB e i relativi valori e come modificare o aggiornare questi valori.

I parametri possono essere dinamici o statici. Quando modifichi un parametro dinamico e salvi il gruppo di parametri del cluster, la modifica viene applicata immediatamente, indipendentemente

dall'impostazione `Apply Immediately`. Quando modifichi un parametro statico e salvi il gruppo di parametri del cluster, la modifica del parametro viene applicata solo dopo che avrai riavviato manualmente le istanze.

Visualizzazione dei parametri di un gruppo di parametri del cluster Amazon DocumentDB

Puoi visualizzare i parametri di un cluster Amazon DocumentDB e i relativi valori utilizzando il Console di gestione AWS pulsante or. AWS CLI

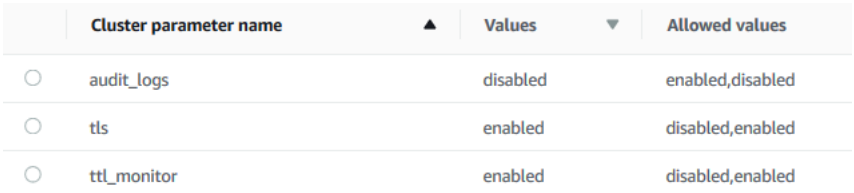
Using the Console di gestione AWS

Per visualizzare i dettagli di un gruppo di parametri del cluster

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nel riquadro Parameter groups (Gruppi di parametri), scegli il nome del gruppo di parametri del cluster di cui desideri visualizzare i dettagli.
4. La pagina risultante mostra i seguenti valori per ciascun parametro: nome del parametro, valore corrente, valori consentiti, se il parametro è modificabile, tipo di applicazione, tipo di dati e descrizione.



	Cluster parameter name ▲	Values ▼	Allowed values
○	audit_logs	disabled	enabled,disabled
○	tls	enabled	disabled,enabled
○	ttl_monitor	enabled	disabled,enabled

Using the AWS CLI

Per visualizzare i parametri e i valori di un gruppo di parametri del cluster, utilizza l'operazione `describe-db-cluster-parameters` con i parametri seguenti.

- **`--db-cluster-parameter-group-name`**: obbligatorio. Il nome del gruppo di parametri del cluster di cui ottenere un elenco di parametri dettagliato.
- **`--source`**— Facoltativo. Se fornito, restituisce solo i parametri per una determinata origine. Le origini dei parametri possono essere `engine-default`, `system` o `user`.

Example

Il codice seguente elenca i parametri e i relativi valori per il gruppo di parametri `custom3-6-param-grp`. Per ulteriori informazioni sul gruppo di parametri, ometti la riga `--query`. Per informazioni su tutti i gruppi di parametri, ometti la riga `--db-cluster-parameter-group-name`.

Per Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameters \
   --db-cluster-parameter-group-name custom3-6-param-grp \
   --query 'Parameters[*].[ParameterName,ParameterValue]'
```

Per Windows:

```
aws docdb describe-db-cluster-parameters ^
   --db-cluster-parameter-group-name custom3-6-param-grp ^
   --query 'Parameters[*].[ParameterName,ParameterValue]'
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
[
    [
        "audit_logs",
        "disabled"
    ],
    [
        "tls",
        "enabled"
    ],
    [
        "ttl_monitor",
        "enabled"
    ]
]
```

Modifica dei parametri di un gruppo di parametri del cluster Amazon DocumentDB

È possibile modificare i parametri di un gruppo di parametri utilizzando o. Console di gestione AWS AWS CLI

Using the Console di gestione AWS

Per aggiornare i parametri di un gruppo di parametri del cluster

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nel riquadro Parameter groups (Gruppi di parametri) scegliere il gruppo di parametri del cluster di cui si desidera aggiornare i parametri.
4. La pagina risultante mostra i parametri e i relativi dettagli corrispondenti per questo gruppo di parametri del cluster. Selezionare un parametro da aggiornare.
5. In alto a destra della pagina, scegliere Edit (Modifica) per modificare il valore del parametro. Per ulteriori informazioni sui tipi di parametri del cluster, consulta Riferimento ai parametri del cluster Amazon DocumentDB.
6. Apportare la modifica, quindi scegliere Modify cluster parameter (Modifica parametro cluster) per salvare le modifiche. Per annullare le modifiche, selezionare Cancel (Annulla).

Using the AWS CLI

Per modificare i parametri di un gruppo di parametri del cluster, utilizzare l'operazione `modify-db-cluster-parameter-group` con i parametri seguenti.

- **`--db-cluster-parameter-group-name`**: obbligatorio. Il nome del gruppo di parametri del cluster che stai modificando.

- **`--parameters`**: obbligatorio. Il parametro o i parametri che stai modificando. Ogni voce del parametro deve includere:
  - **`ParameterName`**— Il nome del parametro che stai modificando.
  - **`ParameterValue`**— Il nuovo valore per questo parametro.
  - **`ApplyMethod`**— Come si desidera che vengano applicate le modifiche a questo parametro. I valori consentiti sono `immediate` e `pending-reboot`.

    > **Note**
    >
    > I parametri con `ApplyType` per `static` devono avere `ApplyMethod` per `pending-reboot`.

Per modificare i valori dei parametri di un gruppo di parametri del cluster (AWS CLI)

L'esempio seguente cambia il parametro `tls`.

1. Elenca i parametri e i relativi valori di **`sample-parameter-group`**.

   Per Linux, macOS o Unix:

   ```
   aws docdb describe-db-cluster-parameters \
       --db-cluster-parameter-group-name sample-parameter-group
   ```

   Per Windows:

   ```
   aws docdb describe-db-cluster-parameters ^
       --db-cluster-parameter-group-name sample-parameter-group
   ```

   L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

   ```
   {
       "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. Modifica il parametro **`tls`** in modo che il relativo valore sia **`disabled`**. Non puoi modificare `ApplyMethod` perché `ApplyType` è `static`.

   Per Linux, macOS o Unix:

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 "ParameterName"=tls,ParameterValue=disabled,ApplyMethod=pending-reboot"
```

   Per Windows:

```
aws docdb modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name sample-parameter-group ^
    --parameters "ParameterName=tls,ParameterValue=disabled,ApplyMethod=pending-
reboot"
```

   L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "DBClusterParameterGroupName": "sample-parameter-group"
}
```

3. Attendi almeno 5 minuti.
4. Elenca i valori dei parametri di **`sample-parameter-group`**.

   Per Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

   Per Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

   L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "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"
        }
    ]
}
```

Comprendere gli endpoint di Amazon DocumentDB

Puoi usare gli endpoint Amazon DocumentDB (con compatibilità con MongoDB) per connetterti a un cluster o a un'istanza. Amazon DocumentDB ha tre diversi tipi di endpoint, ciascuno con un proprio scopo.

Argomenti

- Individuazione degli endpoint di un cluster
- Individuazione dell'endpoint di un'istanza
- Connessione agli endpoint

Endpoint del cluster

Un endpoint del cluster è un endpoint per un cluster Amazon DocumentDB che si connette all'istanza primaria corrente del cluster. Ogni cluster Amazon DocumentDB ha un singolo endpoint del cluster e un'istanza principale. Nel caso di un failover, l'endpoint del cluster viene rimappato alla nuova istanza primaria.

Endpoint di lettura

Un endpoint reader è un endpoint per un cluster Amazon DocumentDB che si connette a una delle repliche disponibili per quel cluster. Ogni cluster Amazon DocumentDB dispone di un endpoint di lettura. Se esiste più di una replica, l'endpoint del lettore indirizza ogni richiesta di connessione a una delle repliche di Amazon DocumentDB.

Endpoint dell'istanza

Per endpoint dell'istanza si intende un endpoint che si connette a un'istanza specifica. Ogni istanza in un cluster, a prescindere che sia primaria o di replica, ha un proprio endpoint dell'istanza esclusivo. Si consiglia di non utilizzare gli endpoint dell'istanza nell'applicazione. Questo perché possono variare ruoli in caso di un failover, richiedendo in tal modo modifiche del codice nell'applicazione.

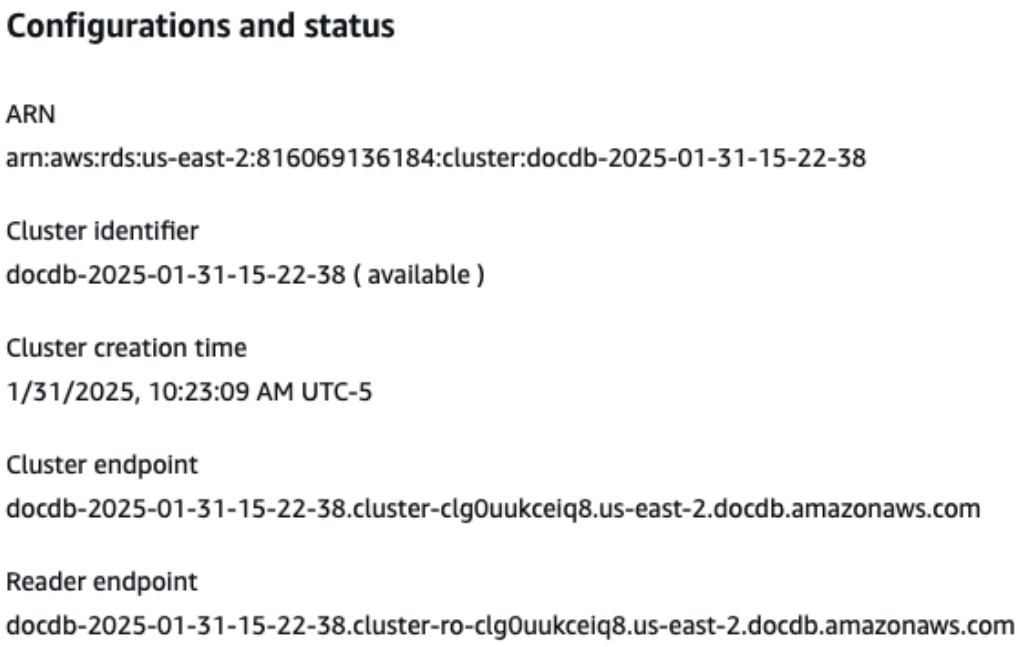
Individuazione degli endpoint di un cluster

Puoi trovare l'endpoint del cluster e l'endpoint reader di un cluster utilizzando la console Amazon DocumentDB o. AWS CLI

Using the Console di gestione AWS

Per trovare gli endpoint di un cluster utilizzando la console:

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.
3. Dall'elenco dei cluster scegli il nome del cluster desiderato.
4. Nella pagina dei dettagli del cluster, seleziona la scheda Configurazione. Nella sezione Configurazioni e stato, troverai l'endpoint Cluster e l'endpoint Reader.



5. Per connetterti a questo cluster, seleziona la scheda Connettività e sicurezza. Individua la stringa di connessione per la mongo shell e la stringa di connessione che può essere utilizzata nel codice dell'applicazione per connettersi al cluster.



Using the AWS CLI

Per trovare gli endpoint del cluster e del lettore per il cluster utilizzando AWS CLI, esegui il `describe-db-clusters` comando con questi parametri.

Parameters

- **`--db-cluster-identifier`**—Facoltativo. Specifica il cluster per il quale restituire l'endpoint. Se omesso, restituisce gli endpoint per un massimo di 100 cluster.
- **`--query`**—Facoltativo. Specifica i campi da visualizzare. È utile per ridurre la quantità di dati da visualizzare per trovare gli endpoint. Se omesso, vengono restituite tutte le informazioni su un cluster.
- **`--region`**—Facoltativo. Il parametro `--region` consente di specificare la regione a cui applicare il comando. Se omesso, viene usata la regione predefinita.

Example

L'esempio seguente restituisce l'endpoint `DBClusterIdentifier` (endpoint del cluster) e `ReaderEndpoint` per `sample-cluster`.

Per Linux, macOS o Unix:

```
aws docdb describe-db-clusters \
   --region us-east-1 \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'
```

Per Windows:

```
aws docdb describe-db-clusters ^
   --region us-east-1 ^
   --db-cluster-identifier sample-cluster ^
   --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
[
  [
     "sample-cluster",
     27017,
```

```
    "sample-cluster.cluster-corlsfccjozr.us-east-1.docdb.amazonaws.com",
    "sample-cluster.cluster-ro-corlsfccjozr.us-east-1.docdb.amazonaws.com"
  ]
]
```

Dopo aver installato l'endpoint del cluster, puoi connetterti al cluster con `mongo` o `mongodb`. Per ulteriori informazioni, consulta Connessione agli endpoint.

Individuazione dell'endpoint di un'istanza

Puoi trovare l'endpoint per un'istanza utilizzando la console Amazon DocumentDB o il. AWS CLI

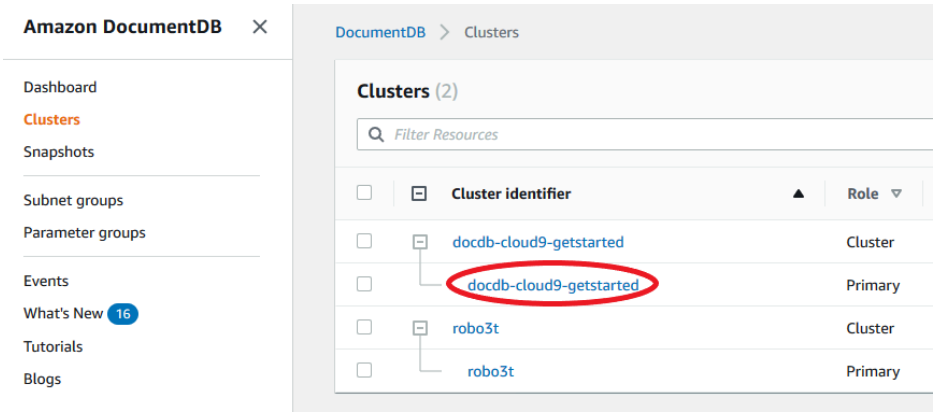
Using the Console di gestione AWS

Per trovare l'endpoint dell'istanza con la console

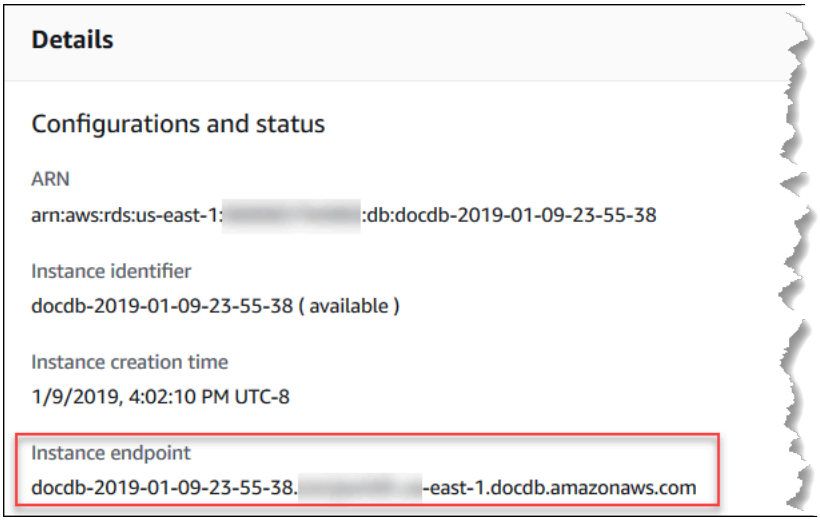
1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.

> **Tip**
>
> Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

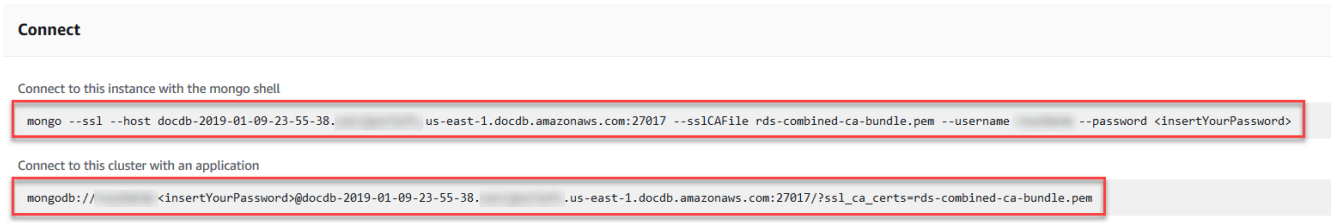
3. Nella casella di navigazione Clusters, vedrai la colonna Cluster Identifier. Le tue istanze sono elencate in cluster, in modo simile alla schermata seguente.



4. Seleziona la casella a sinistra dell'istanza che ti interessa.
5. Scorri la sezione Details (Dettagli) per individuare l'endpoint dell'istanza.



6. Per connetterti a questa istanza, scorri fino alla sezione Connect (Connetti). Individua la stringa di connessione per la shell `mongo` e una stringa di connessione che può essere utilizzata nel codice dell'applicazione per connetterti all'istanza.



Using the AWS CLI

Per trovare l'endpoint dell'istanza utilizzando il AWS CLI, esegui il comando seguente con questi argomenti.

Arguments (Argomenti)

- **`--db-instance-identifier`**—Facoltativo. Specifica l'istanza per la quale restituire l'endpoint. Se omesso, restituisce l'endpoint per un massimo di 100 istanze.
- **`--query`**—Facoltativo. Specifica i campi da visualizzare. È utile per ridurre la quantità di dati da visualizzare per trovare gli endpoint. Se omesso, vengono restituite tutte le informazioni su un'istanza. Il campo `Endpoint` ha tre membri, quindi la sua inclusione nella query, come nell'esempio seguente, restituisce tutti e tre i membri. Se sei interessato solo ad alcuni dei

membri `Endpoint`, sostituisci `Endpoint` nella query con i membri desiderati, come nel secondo esempio.

- **`--region`**—Facoltativo. Il parametro `--region` consente di specificare la regione a cui applicare il comando. Se omesso, viene usata la regione predefinita.

Example

Per Linux, macOS o Unix:

```
aws docdb describe-db-instances \
    --region us-east-1 \
    --db-instance-identifier sample-cluster-instance \
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint]'
```

Per Windows:

```
aws docdb describe-db-instances ^
    --region us-east-1 ^
    --db-instance-identifier sample-cluster-instance ^
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint]'
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
[
    [
        "sample-cluster-instance",
        {
            "Port": 27017,
            "Address": "sample-cluster-instance.corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
            "HostedZoneId": "Z2R2ITUGPM61AM"
        }
    ]
]
```

Riducendo l'output per eliminare `HostedZoneId` del codice puoi modificare la query specificando `Endpoint.Port` e `Endpoint.Address`.

Per Linux, macOS o Unix:

```
aws docdb describe-db-instances \
    --region us-east-1 \
    --db-instance-identifier sample-cluster-instance \
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint.Port,Endpoint.Address]'
```

Per Windows:

```
aws docdb describe-db-instances ^
    --region us-east-1 ^
    --db-instance-identifier sample-cluster-instance ^
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint.Port,Endpoint.Address]'
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
[
    [
        "sample-cluster-instance",
        27017,
        "sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com"
    ]
]
```

Dopo aver installato l'endpoint dell'istanza, puoi connetterti all'istanza con `mongo` o `mongodb`. Per ulteriori informazioni, consulta Connessione agli endpoint.

Connessione agli endpoint

Quando hai l'endpoint del cluster o dell'istanza, puoi connettersi a esso utilizzando la shell `mongo` o una stringa di connessione.

Connessione tramite la shell mongo

Utilizza la struttura seguente per costruire la stringa necessaria per connetterti al cluster o all'istanza utilizzando la shell `mongo`:

```
mongo \
    --ssl \
    --host Endpoint:Port \
    --sslCAFile global-bundle.pem \
```

```
    --username UserName \
    --password Password
```

esempi di **`conchiglie mongo`**

Connessione a un cluster.

```
mongo \
    --ssl \
    --host sample-cluster.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

Connessione a un'istanza:

```
mongo \
    --ssl \
    --host sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

Connessione tramite una stringa di connessione

Utilizza la struttura seguente per costruire la stringa di connessione necessaria per connetterti al cluster o all'istanza.

```
mongodb://UserName:Password@endpoint:port?replicaSet=rs0&ssl_ca_certs=global-
bundle.pem
```

Esempi di stringa di connessione

Connessione a un cluster.

```
mongodb://UserName:Password@sample-cluster.cluster-corlsfccjozr.us-
east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=global-bundle.pem
```

Connessione a un'istanza:

```
mongodb://UserName:Password@sample-cluster-instance.cluster-corlsfccjozr.us-
east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=global-bundle.pem
```

Informazioni su Amazon DocumentDB Amazon Resource Names (ARN)

Le risorse che AWS crei sono identificate in modo univoco con un Amazon Resource Name (ARN). Per alcune operazioni con Amazon DocumentDB (con compatibilità con MongoDB), devi identificare in modo univoco una risorsa Amazon DocumentDB specificandone l'ARN. Ad esempio, quando aggiungi un tag a una risorsa, devi fornire l'ARN della risorsa.

Argomenti

- Creazione di un ARN per una risorsa Amazon DocumentDB
- Individuazione dell'ARN di una risorsa Amazon DocumentDB

Creazione di un ARN per una risorsa Amazon DocumentDB

Puoi creare un ARN per una risorsa Amazon DocumentDB utilizzando la seguente sintassi. Amazon DocumentDB condivide il formato degli ARN di Amazon Relational Database Service (Amazon RDS). Gli ARN di Amazon DocumentDB contengono e non contengono`rds`. `docdb`

`arn:aws:rds:`*`region`*`:`*`account_number`*`:`*`resource_type`*`:`*`resource_id`*

Nome della regione	Regione	Zone di disponibilità (elaborazione)
Stati Uniti orientali (Ohio)	`us-east-2`	3
Stati Uniti orientali (Virginia settentrionale)	`us-east-1`	6
Stati Uniti occidentali (Oregon)	`us-west-2`	4

Nome della regione	Regione	Zone di disponibilità (elaborazione)
Africa (Città del Capo)	af-south-1	3
Sud America (San Paolo)	sa-east-1	3
Asia Pacifico (Hong Kong)	ap-east-1	3
Asia Pacifico (Hyderabad)	ap-south-2	3
Asia Pacifico (Malesia)	ap-southeast-5	3
Asia Pacifico (Mumbai)	ap-south-1	3
Asia Pacifico (Osaka)	ap-northeast-3	3
Asia Pacifico (Seul)	ap-northeast-2	4
Asia Pacifico (Singapore)	ap-southeast-1	3
Asia Pacifico (Sydney)	ap-southeast-2	3
Asia Pacifico (Giacarta)	ap-southeast-3	3
Asia Pacifico (Melbourne)	ap-southeast-4	3
Asia Pacifico (Thailandia)	ap-southeast-7	3
Asia Pacifico (Tokyo)	ap-northeast-1	3

Nome della regione	Regione	Zone di disponibilità (elaborazione)
Canada (Centrale)	`ca-central-1`	3
Canada occidentale (Calgary)	`ca-west-1`	3
Regione Cina (Pechino)	`cn-north-1`	3
Cina (Ningxia)	`cn-northwest-1`	3
Europa (Francoforte)	`eu-central-1`	3
Europa (Zurigo)	`eu-central-2`	3
Europa (Irlanda)	`eu-west-1`	3
Europa (Londra)	`eu-west-2`	3
Europa (Milano)	`eu-south-1`	3
Europa (Parigi)	`eu-west-3`	3
Europa (Spagna)	`eu-south-2`	3
Europa (Stoccolma)	`eu-north-1`	3
Messico (centrale)	`mx-central-1`	3
Medio Oriente (Emirati Arabi Uniti)	`me-central-1`	3
Israele (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**
>
> L'architettura Amazon DocumentDB separa storage ed elaborazione. Per il livello di storage, Amazon DocumentDB replica sei copie dei dati su tre zone di AWS disponibilità (AZ). Le AZ elencate nella tabella precedente rappresentano il numero di AZ che puoi utilizzare in una determinata regione per effettuare il provisioning delle istanze di calcolo. Ad esempio, se stai lanciando un cluster Amazon DocumentDB in ap-northeast-1, lo storage verrà replicato in sei modi su tre AZ, ma le tue istanze di calcolo saranno disponibili solo in due AZ.

La tabella seguente mostra il formato da utilizzare per creare un ARN per una particolare risorsa Amazon DocumentDB. Amazon DocumentDB condivide il formato degli ARN Amazon RDS. Gli ARN di Amazon DocumentDB contengono e non contengonords. docdb

Tipo di risorsa	Formato ARN/Esempio
Istanza (db)	arn:aws:rds: *region*:*account_number* :db:*resource_id* 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*
Gruppo di parametri del cluster (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*
Gruppo di sicurezza (secgrp)	arn:aws:rds: *region*:*account_number* :secgrp:*resource_id*

Tipo di risorsa	Formato ARN/Esempio
	`arn:aws:rds:us-east-1:` *1234567890* `:secgrp:`*sample-public-secgrp*
Snapshot del cluster (`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*
Gruppo di sottoreti (`subgrp`)	`arn:aws:rds:` *region*`:`*account_number* `:subgrp:`*resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:subgrp:`*sample-subnet-10*

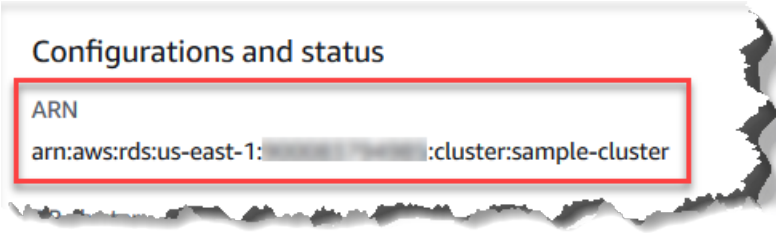
Individuazione dell'ARN di una risorsa Amazon DocumentDB

Puoi trovare l'ARN di una risorsa Amazon DocumentDB utilizzando Console di gestione AWS o il. AWS CLI

Using the Console di gestione AWS

Per trovare un ARN utilizzando la console, accedi alla risorsa di cui desideri l'ARN e visualizzane i relativi dettagli.

Ad esempio, è possibile ottenere l'ARN per un cluster selezionando la scheda Configurazione nella pagina dei dettagli del cluster. L'ARN è disponibile nella sezione Configurazioni e stato, come mostrato nella schermata seguente.



Using the AWS CLI

Per ottenere un ARN utilizzando il AWS CLI per una particolare risorsa Amazon DocumentDB, usa `describe` l'operazione per quella risorsa. La tabella seguente mostra ogni AWS CLI operazione e la proprietà ARN utilizzata con l'operazione per ottenere un ARN.

AWS CLI Comando	Proprietà ARN
`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`
`describe-db-cluster-snapshots`	`DBClusterSnapshotArn`
`describe-db-subnet-groups`	`DBSubnetGroupArn`

Example- Ricerca dell'ARN per un cluster

L' AWS CLI operazione seguente trova l'ARN per il cluster. `sample-cluster`

Per Linux, macOS o Unix:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterArn'
```

Per Windows:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterArn'
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
[
    "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster"
]
```

Example- Ricerca degli ARN per più gruppi di parametri

Per Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameter-groups \
   --query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

Per Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
   --query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
[
    "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"
]
```

Etichettatura delle risorse di Amazon DocumentDB

Puoi utilizzare i tag Amazon DocumentDB (con compatibilità con MongoDB) per aggiungere metadati alle tue risorse Amazon DocumentDB. Questi tag possono essere utilizzati con le policy AWS Identity and Access Management (IAM) per gestire l'accesso alle risorse di Amazon DocumentDB e per controllare quali azioni possono essere applicate alle risorse. Puoi utilizzare questi tag anche per tenere traccia dei costi raggruppando le spese per risorse con tag simili.

Puoi taggare le seguenti risorse Amazon DocumentDB:

- Cluster
- Istanze

- Snapshot
- Snapshot cluster
- Gruppi di parametri
- Gruppi di parametri di cluster
- Gruppi di sicurezza
- Gruppi di sottoreti

Panoramica dei tag di risorsa di Amazon DocumentDB

Un tag Amazon DocumentDB è una coppia nome-valore che definisci e associ a una risorsa Amazon DocumentDB. Il nome viene definito chiave. L'indicazione di un valore per la chiave è un'operazione facoltativa. Puoi utilizzare i tag per assegnare informazioni arbitrarie a una risorsa Amazon DocumentDB. Una chiave tag potrebbe essere impiegata, ad esempio, per definire una categoria e il valore di tag potrebbe essere un elemento di tale categoria. Ad esempio, è possibile definire una chiave di tag `project` e un valore di tag di`Salix`, a indicare che la risorsa Amazon DocumentDB è assegnata al progetto Salix. Puoi anche utilizzare i tag per designare le risorse di Amazon DocumentDB da utilizzare per test o produzione utilizzando una chiave come `environment=test` o. `environment=production` Ti consigliamo di utilizzare un set coerente di chiavi di tag per semplificare il monitoraggio dei metadati associati alle risorse di Amazon DocumentDB.

Puoi utilizzare i tag per organizzare la AWS fattura in modo da rispecchiare la tua struttura di costi. A tale scopo, registrati per ricevere la Account AWS fattura con i valori chiave dell'etichetta inclusi. Per visualizzare il costo delle risorse combinate, puoi organizzare le informazioni di fatturazione in base alle risorse con gli stessi valori di chiave di tag. Puoi ad esempio applicare tag a numerose risorse con un nome di applicazione specifico, quindi organizzare le informazioni di fatturazione per visualizzare il costo totale dell'applicazione in più servizi. Per ulteriori informazioni, consulta Utilizzo dei tag per l'allocazione dei costi nella AWS Guida per l'utente di Billing and Cost Management.

Ogni risorsa Amazon DocumentDB ha un set di tag che contiene tutti i tag assegnati a quella risorsa. Un set di tag può contenere fino a 10 tag ma può anche essere vuoto. Se aggiungi un tag a una risorsa Amazon DocumentDB che ha la stessa chiave di un tag esistente sulla risorsa, il nuovo valore sovrascrive il vecchio valore.

AWS non applica alcun significato semantico ai tag; i tag vengono interpretati rigorosamente come stringhe di caratteri. Amazon DocumentDB può impostare tag su un'istanza o altre risorse Amazon DocumentDB, a seconda delle impostazioni utilizzate al momento della creazione della risorsa. Ad

esempio, Amazon DocumentDB potrebbe aggiungere un tag che indica che un'istanza è destinata alla produzione o al test.

Puoi aggiungere un tag a una snapshot, tuttavia questo raggruppamento non sarà indicato riportato nella fattura.

Puoi usare Console di gestione AWS o AWS CLI per aggiungere, elencare ed eliminare tag sulle risorse di Amazon DocumentDB. Quando utilizzi AWS CLI, devi fornire l'Amazon Resource Name (ARN) per la risorsa con cui desideri lavorare. Per ulteriori informazioni sugli ARN di Amazon DocumentDB, consulta. Informazioni su Amazon DocumentDB Amazon Resource Names (ARN)

Vincoli relativi ai tag

I seguenti vincoli si applicano ai tag Amazon DocumentDB:

- Numero massimo di tag per risorsa: 10
- Lunghezza massima della chiave: 128 caratteri Unicode
- Lunghezza massima del valore: 256 caratteri Unicode
- Caratteri validi per Chiave e Valore: lettere maiuscole e minuscole nel set di UTF-8 caratteri, cifre, spazio e i seguenti caratteri: `_ . : / = + -` e `@` (espressione regolare Java:) `"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"`
- I valori e le chiavi dei tag rispettano la distinzione tra maiuscole e minuscole.
- Il prefisso `aws:` non può essere utilizzato per chiavi o valori dei tag, ma è riservato per AWS.

Aggiungere e aggiornare tag su una risorsa Amazon DocumentDB

Puoi aggiungere fino a 10 tag a una risorsa utilizzando Console di gestione AWS o il AWS CLI.

Using the Console di gestione AWS

Il processo di aggiunta di un tag a una risorsa è quasi identico, indipendentemente dal tipo di risorsa. In questo esempio, il tag viene aggiunto a un cluster.

Per aggiungere o aggiornare i tag in un cluster con la console

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel riquadro di navigazione scegli clusters (cluster).

3. Seleziona il nome del cluster a cui aggiungere i tag.
4. Scorri verso il basso fino alla sezione Tags (Tag) e scegli Edit (Modifica).
5. Per ogni tag da aggiungere alla risorsa, procedi nel seguente modo:

    a. Per aggiungere un nuovo tag, immetti il nome del tag nella casella Key (Chiave). Per modificare il valore di un tag, individua il nome del tag nella colonna Key (Chiave).

    b. Per assegnare al tag un valore nuovo o aggiornato, inserisci un valore per il tag nella casella Value (Valore).

    c. Se desideri aggiungere altri tag, scegli Add (Aggiungi). Altrimenti, al termine, scegli Save (Salva).

Using the AWS CLI

Il processo di aggiunta di un tag a una risorsa è quasi identico, indipendentemente dal tipo di risorsa. In questo esempio, vengono aggiunti tre tag a un cluster. Il secondo tag `key2` non contiene valori.

Usa l' AWS CLI operazione `add-tags-to-resource` con questi parametri.

Parameters

- **`--resource-name`**—L'ARN della risorsa Amazon DocumentDB a cui desideri aggiungere i tag.
- **`--tags`**—Un elenco dei tag (coppia chiave-valore) che desideri aggiungere a questa risorsa nel formato. `Key=`*`key-name`*`,Value=`*`tag-value`*

Example

Per Linux, macOS o 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
```

Per 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
```

L'operazione `add-tags-to-resource` non produce alcun output. Per visualizzare i risultati, utilizza l'operazione `list-tags-for-resource`.

Elencare i tag su una risorsa Amazon DocumentDB

Puoi usare Console di gestione AWS o the AWS CLI per ottenere un elenco dei tag per una risorsa Amazon DocumentDB.

Using the Console di gestione AWS

Il processo per elencare i tag in una risorsa è quasi identico, indipendentemente dal tipo di risorsa a cui si aggiunge il tag. In questo esempio, vengono elencati i tag per un cluster.

Per elencare i tag in un cluster con la console

1. Apri la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel riquadro di navigazione scegli clusters (cluster).
3. Seleziona il nome del cluster di cui desideri elencare i tag.
4. Per visualizzare un elenco dei tag in questa risorsa, scorri la sezione Tags (Tag).

Using the AWS CLI

Il processo per elencare i tag in una risorsa è quasi identico, indipendentemente dal tipo di risorsa di cui si elencano i tag. In questo esempio, vengono elencati i tag in un cluster.

Usa l' AWS CLI operazione `list-tags-for-resource` con questi parametri.

Parameters

- **`--resource-name`**: obbligatorio. L'ARN della risorsa Amazon DocumentDB per cui desideri elencare i tag.

Example

Per Linux, macOS o Unix:

```
aws docdb list-tags-for-resource \
```

```
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster
```

Per Windows:

```
aws docdb list-tags-for-resource ^
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "TagList": [
        {
            "Key": "key1",
            "Value": "value1"
        },
        {
            "Key": "key2",
            "Value": ""
        },
        {
            "Key": "key3",
            "Value": "value3"
        }
    ]
}
```

Rimozione di tag da una risorsa Amazon DocumentDB

Puoi usare Console di gestione AWS o AWS CLI per rimuovere i tag dalle risorse di Amazon DocumentDB.

Using the Console di gestione AWS

Il processo di rimozione dei tag da una risorsa è quasi identico, indipendentemente dal tipo di risorsa. In questo esempio, vengono rimossi i tag da un cluster.

Per rimuovere tag da un cluster mediante la console

1. Apri la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel riquadro di navigazione scegli clusters (cluster).

3. Seleziona il nome del cluster da cui desideri rimuovere i tag.
4. Scorri verso il basso fino alla sezione Tags (Tag) e scegli Edit (Modifica).
5. Se desideri rimuovere tutti i tag da questa risorsa, scegli Remove all (Rimuovi tutti). Altrimenti, per ogni tag da rimuovere dalla risorsa, procedi nel seguente modo:

   a. Individua il nome del tag nella colonna Key (Chiave).

   b. Scegli Remove (Rimuovi) sulla stessa riga della chiave del tag.

   c. Al termine, scegli Save (Salva).

Using the AWS CLI

Il processo di rimozione di un tag da una risorsa è quasi identico, indipendentemente dal tipo di risorsa da cui viene rimosso il tag. In questo esempio, viene rimosso un tag da un cluster.

Usa l' AWS CLI operazione `remove-tags-from-resource` con questi parametri.

- **`--resource-name`**: obbligatorio. L'ARN della risorsa Amazon DocumentDB da cui desideri rimuovere i tag.
- **`--tag-keys`**: obbligatorio. Un elenco di chiavi dei tag che desideri rimuovere da questa risorsa.

Example

Per Linux, macOS o Unix:

```
aws docdb remove-tags-from-resource \
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tag-keys key1 key3
```

Per Windows:

```
aws docdb remove-tags-from-resource ^
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tag-keys key1 key3
```

L'operazione `removed-tags-from-resource` non produce alcun output. Per visualizzare i risultati, utilizza l'operazione `list-tags-for-resource`.

Manutenzione di Amazon DocumentDB

Amazon DocumentDB esegue periodicamente due tipi di manutenzione:

- La manutenzione del cluster aggiorna il motore del database. Gli aggiornamenti del motore includono correzioni di sicurezza, correzioni di bug, nuove funzionalità e altri miglioramenti del motore.
- La manutenzione dell'istanza aggiorna il sistema operativo (OS) dell'istanza.

Le patch del motore e gli aggiornamenti del sistema operativo utilizzano le stesse tre categorie del ciclo di vita (facoltativa , obbligatoria e forzata) con lo stesso comportamento di notifica e applicazione per ciascuna categoria. Le versioni del motore hanno anche una quarta categoria: le versioni secondarie, alle quali è possibile eseguire l'aggiornamento manualmente. Le categorie sono:

- Facoltativo: contiene miglioramenti non critici. Nessuna data di applicazione automatica e nessuna notifica AHD; applica quando preferisci. (Per gli aggiornamenti del sistema operativo, puoi abbonarti per `RDS-EVENT-0230` ricevere una notifica quando ne sarà disponibile uno.)
- Obbligatorio: contiene correzioni di sicurezza e altre correzioni critiche. Riceverai una notifica tramite Health Dashboard (AHD) ed e-mail. Un'azione richiesta si applica automaticamente durante la finestra di manutenzione del cluster o dell'istanza successiva. `AutoAppliedAfterDate` Puoi differire modificando la finestra di manutenzione prima di tale data.
- Forzato: una correzione rara e molto critica. Auto-applies fuori dalla finestra di manutenzione dopo la sua `ForcedApplyDate` scadenza. Amazon DocumentDB indica un'azione forzata solo quando nessun'altra opzione è disponibile.
- Versione secondaria (solo versioni del motore): una versione numerata del motore che si aggiunge a una versione principale (ad esempio). `5.0.1` User-driven: si esegue l'upgrade modificando la versione del motore del cluster. Non si applica mai automaticamente; nessuna notifica AHD. Le versioni secondarie non vengono pubblicate per le versioni principali precedenti alla 5.0.

Le patch del motore vengono rilasciate in un'unica categoria (opzionale, obbligatoria o forzata) e vi rimangono. Progresso degli aggiornamenti del sistema operativo: la maggior parte viene avviata come facoltativa e, se non applicata, diventa obbligatoria e infine forzata. La tempistica esatta dipende dalla patch ed è pubblicata nella notifica AHD e nei campi della data restituiti da `describe-pending-maintenance-actions` (vediApplica le date). Le note di rilascio di Amazon DocumentDB utilizzano questi nomi di categoria quando annunciano le modifiche al motore.

L'applicazione di qualsiasi patch del motore porta il cluster offline per un breve periodo. Il resto di questo argomento illustra come funzionano le finestre di manutenzione, come trovare il lavoro in sospeso, come applicare le patch del motore e le versioni secondarie, come funzionano gli aggiornamenti del sistema operativo e la gestione speciale dei cluster globali.

Argomenti

- Azioni di manutenzione per Amazon DocumentDB
- Numerazione delle versioni del motore
- Gestione delle finestre di manutenzione di Amazon DocumentDB
- Notifiche per le patch del motore Amazon DocumentDB
- Visualizzazione delle azioni di manutenzione di Amazon DocumentDB in sospeso
- Aggiornamenti del motore Amazon DocumentDB
- Aggiornamenti a versioni secondarie
- Aggiornamenti del sistema operativo Amazon DocumentDB
- User-initiated aggiornamenti
- Applicazione di patch ai cluster globali

Azioni di manutenzione per Amazon DocumentDB

Le seguenti azioni di manutenzione si applicano ai cluster Amazon DocumentDB:

- `system-update`— Aggiorna la patch del motore per il cluster Amazon DocumentDB. Per ulteriori informazioni, consulta Aggiornamenti del motore Amazon DocumentDB.
- `os-upgrade`— Aggiorna i sistemi operativi di tutte le istanze DB nel cluster Amazon DocumentDB, utilizzando aggiornamenti continui. Per ulteriori informazioni, consulta Aggiornamenti del sistema operativo Amazon DocumentDB.

Le seguenti azioni di manutenzione si applicano alle istanze Amazon DocumentDB:

- `system-update`— Aggiorna il sistema operativo dell'istanza Amazon DocumentDB. Ti consigliamo invece di utilizzare l'azione di `os-upgrade` manutenzione a livello di cluster. Per ulteriori informazioni, consulta Aggiornamenti del sistema operativo Amazon DocumentDB.

Numerazione delle versioni del motore

Amazon DocumentDB utilizza due identificatori di versione separati:

- Versione del motore: un numero composto da tre parti nel modulo *major*.*major*.*minor* (ad esempio, o). `5.0.0` `5.0.1` Le prime due parti (`5.0`) sono la versione di compatibilità con MongoDB; la terza parte è la versione secondaria, incrementata quando Amazon DocumentDB pubblica una versione secondaria contenente correzioni di bug e miglioramenti continui. Questa è la versione specificata durante la creazione o l'aggiornamento di un cluster.
- Versione della patch del motore: un numero separato in tre parti nel modulo *major*`.0.`*patch* (ad esempio`3.0.17983`) che identifica il livello di patch applicato al cluster. La cifra centrale è sempre. `0` Le versioni delle patch contengono correzioni critiche di sicurezza e stabilità.

Quando la versione della patch del motore viene visualizzata da sola, è possibile determinarla in base al relativo prefisso, come mostrato nella tabella seguente.

Prefisso della versione della patch del motore	Versione del motore Amazon DocumentDB
`1.0.`*x*	3.6
`2.0.`*x*	4.0
`3.0.`*x*	5.0
`4.0.`*x*	8.0

Per verificare la versione della patch in esecuzione sul cluster, connettiti ed `db.runCommand({getEngineVersion: 1})` esegui.

> **Note**
>
> A partire da Amazon DocumentDB 5.0.2 e 8.0.2, `db.runCommand({getEngineVersion: 1})` restituisce la versione del motore e la versione della patch del motore come una singola stringa nel modulo. *engine-version*+*engine-patch-version* Ad esempio, restituisce un cluster su Amazon DocumentDB 5.0.2 e un cluster su `5.0.2+3.0.12345` 8.0.2 restituisce. `8.0.2+4.0.12345` I cluster nelle versioni precedenti restituiscono da soli la versione della patch del motore, ad esempio. `3.0.12345`

Se disponi di script o sistemi di monitoraggio che analizzano l'output di`getEngineVersion`, aggiornali per gestire il formato combinato.

Per l'elenco delle versioni rilasciate delle patch del motore e il contenuto di ciascuna di esse, consultate. Note di rilascio

Gestione delle finestre di manutenzione di Amazon DocumentDB

Ogni cluster e ogni istanza ha una propria finestra di manutenzione settimanale di 30 minuti, ovvero il periodo in cui vengono eseguite le modifiche pianificate e le patch software. La maggior parte degli eventi viene completata entro 30 minuti; quelli più grandi possono durare più a lungo.

Se non scegli una finestra durante la creazione della risorsa, Amazon DocumentDB ne assegna una a caso entro un blocco giornaliero di 8 ore definito per la regione, in un giorno selezionato a caso. Scegli finestre che riducano al minimo l'impatto sulla tua applicazione, ad esempio di sera o nei fine settimana.

Per gli aggiornamenti del motore di database, Amazon DocumentDB utilizza la finestra del cluster, non le finestre delle singole istanze.

La tabella seguente mostra i blocchi temporali predefiniti per regione.

Nome della regione	Regione	Blocco di tempo UTC
Stati Uniti orientali (Ohio)	us-east-2	03:00-11:00
Stati Uniti orientali (Virginia settentrionale)	us-east-1	03:00-11:00
Stati Uniti occidentali (Oregon)	us-west-2	06:00-14:00
Africa (Città del Capo)	af-south-1	03:00 — 11:00
Asia Pacific (Hong Kong)	ap-east-1	06:00-14:00
Asia Pacifico (Hyderabad)	ap-south-2	DALLE 06:30 ALLE 14:30
Asia Pacifico (Malesia)	ap-southeast-5	13:00-21:00

Nome della regione	Regione	Blocco di tempo UTC
Asia Pacifico (Mumbai)	ap-south-1	06:00-14:00
Asia Pacifico (Osaka-Locale)	ap-northeast-3	12:00-20:00
Asia Pacific (Seoul)	ap-northeast-2	13:00-21:00
Asia Pacifico (Singapore)	ap-southeast-1	14:00-22:00
Asia Pacifico (Sydney)	ap-southeast-2	12:00-20:00
Asia Pacifico (Giacarta)	ap-southeast-3	08:00-16:00
Asia Pacifico (Melbourne)	ap-southeast-4	11:00-19:00
Asia Pacifico (Thailandia)	ap-southeast-7	15:00-23:00
Asia Pacifico (Tokyo)	ap-northeast-1	13:00-21:00
Canada (Centrale)	ca-central-1	03:00-11:00
Canada occidentale (Calgary)	ca-west-1	18:00-02:00
Cina (Pechino)	cn-north-1	06:00-14:00
China (Ningxia)	cn-northwest-1	06:00-14:00
Europa (Francoforte)	eu-central-1	21:00-05:00
Europa (Zurigo)	eu-central-2	02:00-10:00
Europa (Irlanda)	eu-west-1	22:00-06:00
Europe (London)	eu-west-2	22:00-06:00
Europe (Milan)	eu-south-1	02:00-10:00
Europa (Parigi)	eu-west-3	23:59-07:29
Europa (Spagna)	eu-south-2	DALLE 02:00 ALLE 10:00

Nome della regione	Regione	Blocco di tempo UTC
Europa (Stoccolma)	eu-north-1	DALLE 04:00 ALLE 12:00
Messico (Centrale)	mx-central-1	03:00-11:00
Medio Oriente (Emirati Arabi Uniti)	me-central-1	DALLE 05:00 ALLE 13:00
Sud America (San Paolo)	sa-east-1	00:00-08:00
Israele (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

Modifica delle finestre di manutenzione di Amazon DocumentDB

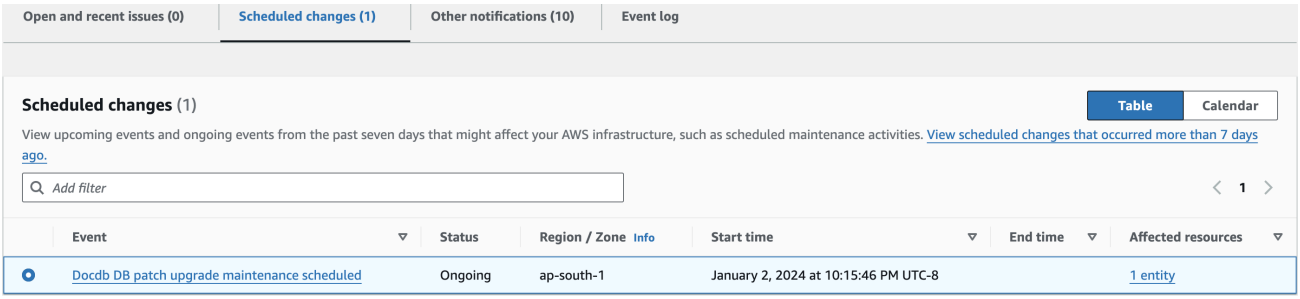
Scegli la finestra di traffico più bassa possibile e modificala nel tempo man mano che i tuoi schemi di traffico cambiano. Il cluster o l'istanza non sono disponibili durante la finestra solo se una modifica del sistema, ad esempio un'operazione di scalabilità dello storage o una modifica della classe di istanza, richiede un'interruzione e solo per il tempo effettivamente necessario a tale modifica.

Per modificare la finestra di manutenzione

- Per un cluster: consulta Modifica di un cluster Amazon DocumentDB
- Per un'istanza: consulta Modifica di un'istanza Amazon DocumentDB.

Notifiche per le patch del motore Amazon DocumentDB

Quando una patch del motore richiesta diventa disponibile in una AWS regione, ogni AWS account con un cluster Amazon DocumentDB interessato in quella regione riceve una notifica tramite Health Dashboard (AHD) e tramite e-mail (inviata all'indirizzo utente principale dell' AWS account). Viene inviata una notifica per versione del motore Amazon DocumentDB interessata. Li puoi trovare in Modifiche pianificate nell'AHD. Ogni notifica elenca i tempi di disponibilità delle patch, la pianificazione dell'applicazione automatica, i cluster interessati e le note di rilascio.



Le patch del motore richieste seguono un unico lead time di circa 30 giorni. Quando una patch diventa disponibile nella tua regione, Amazon DocumentDB invia la notifica sopra descritta. A quel punto, la patch `AutoAppliedAfterDate` è impostata circa 30 giorni dopo. Fino a tale data, la patch rimane in sospeso: puoi applicarla in qualsiasi momento o posticiparla spostando la finestra di manutenzione del cluster a un giorno successivo. Il giorno o dopo`AutoAppliedAfterDate`, la patch si applica automaticamente durante la successiva finestra di manutenzione del cluster.

Ad esempio, una patch richiesta che diventa disponibile il 1° giugno 2026 ha una durata approssimativa `AutoAppliedAfterDate` del 1° luglio 2026. Riceverai la notifica il 1° giugno 2026 e, se non intraprendi alcuna azione, la patch si applica automaticamente durante la prima finestra di manutenzione del cluster, a partire dal 1° luglio 2026.

Dopo aver ricevuto la notifica, hai due opzioni: applicare automaticamente la patch prima della data di applicazione automatica o attendere che si applichi automaticamente durante una prossima finestra di manutenzione (impostazione predefinita). Per applicarla automaticamente, apri la scheda Manutenzione e backup del cluster e cerca la voce relativa al tipo. `system-update`

> **Note**
>
> Lo stato della notifica nell'AHD rimane in corso fino a quando Amazon DocumentDB non rilascia un'altra patch del motore con una nuova versione della patch.
> Dopo l'applicazione della patch, la versione della patch del motore del cluster si aggiorna in modo da corrispondere alla versione nella notifica. Verifica la nuova versione eseguendo`db.runCommand({getEngineVersion: 1})`.

Le patch opzionali e le nuove versioni secondarie non generano notifiche AHD o e-mail. Per tenerne traccia, guarda le note di rilascio di Amazon DocumentDB.

Le patch forzate (la categoria più rara, riservata alle correzioni di sicurezza più critiche) vengono inoltre annunciate tramite AHD ed e-mail. A differenza delle patch obbligatorie, si applicano al di fuori

della finestra di manutenzione, quindi l'esempio di tempistica dell'applicazione automatica riportato sopra non si applica.

Reazione programmatica alle notifiche delle patch

AWS Health si integra con Amazon EventBridge, che consente di creare applicazioni basate sugli eventi su più di 20 destinazioni, tra cui Amazon Simple AWS Lambda Queue Service (SQS). Per reagire in modo programmatico alla disponibilità delle patch del motore, esegui la configurazione in base all'evento. EventBridge `AWS_DOCDB_DB_PATCH_UPGRADE_MAINTENANCE_SCHEDULED` Da lì puoi acquisire i dati degli eventi, generare eventi aggiuntivi, inviare notifiche push tramite il AWS Console Mobile Application o intraprendere qualsiasi altra azione di cui hai bisogno.

Se Amazon DocumentDB annulla una patch (cosa rara), riceverai una notifica AHD e un'e-mail relativa all'annullamento. Usa il codice `AWS_DOCDB_DB_PATCH_UPGRADE_MAINTENANCE_CANCELLED` dell'evento con Amazon EventBridge per gestire questo caso. Per ulteriori informazioni sulla scrittura delle regole, consulta la Amazon EventBridge User Guide.

Visualizzazione delle azioni di manutenzione di Amazon DocumentDB in sospeso

Usa Console di gestione AWS o the AWS CLI per verificare la manutenzione in sospeso per un cluster o un'istanza.

Gli aggiornamenti in sospeso vengono visualizzati con il tipo di azione`system-update`, che copre sia le patch del motore che gli aggiornamenti del sistema operativo.

Quando un aggiornamento è in sospeso, puoi:

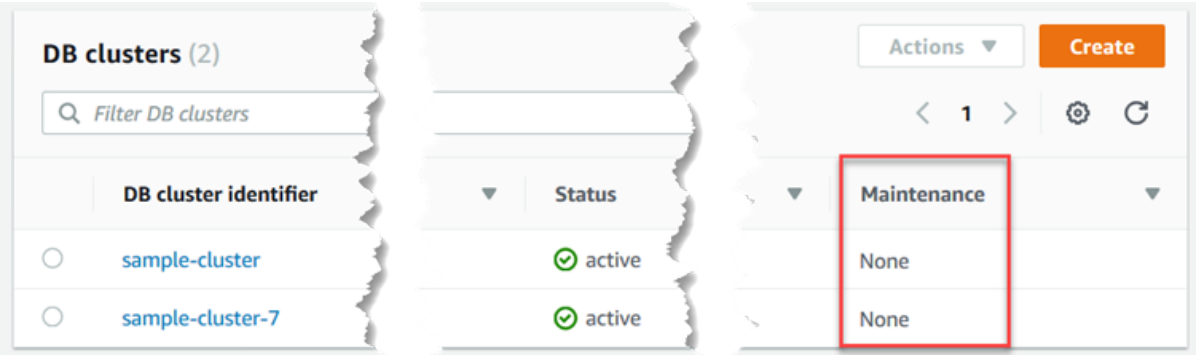
- Applicalo immediatamente.
- Pianificalo per la prossima finestra di manutenzione.
- Rinvialo (solo patch del motore e aggiornamenti del sistema operativo) modificando prima la finestra di manutenzione. `AutoAppliedAfterDate` Trascorsa tale data, l'azione verrà applicata automaticamente durante la successiva finestra di manutenzione. Una volta `ForcedApplyDate` trascorsa, non è possibile alcun ulteriore differimento.

> **Note**
>
> Se non intraprendi alcuna azione, le azioni di manutenzione necessarie, come le patch necessarie al motore, si applicano automaticamente durante una prossima finestra di manutenzione. Le patch opzionali e le versioni secondarie non si applicano mai automaticamente.

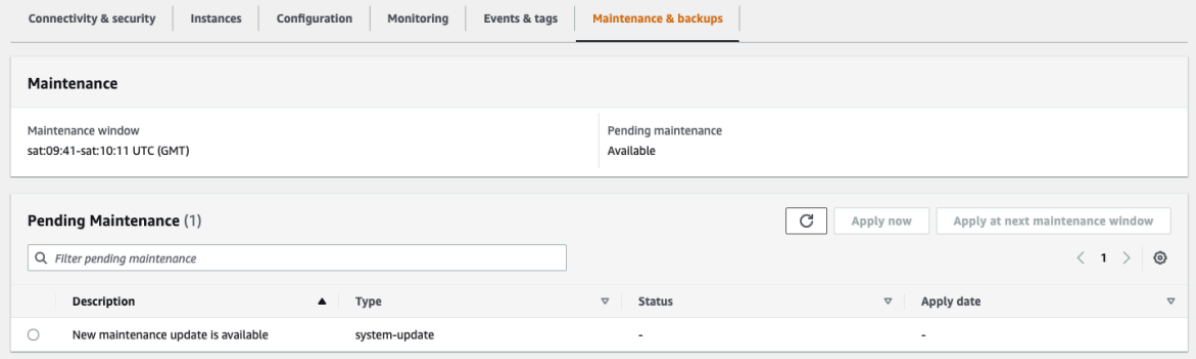
La finestra di manutenzione controlla quando iniziano le operazioni in sospeso, non quanto tempo occorre per completarle.

Using the Console di gestione AWS

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.
3. La colonna Manutenzione del cluster mostra Available, Required o Next Window quando un aggiornamento è in sospeso.



4. Apri il cluster, quindi scegli Manutenzione e backup per visualizzare gli elementi di manutenzione in sospeso e intervenire su di essi.



Using the AWS CLI

Esegui `describe-pending-maintenance-actions` per vedere cosa è in sospeso. L'esempio seguente mostra un account senza azioni in sospeso.

```
aws docdb describe-pending-maintenance-actions
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "PendingMaintenanceActions": []
}
```

Un account con un'azione in sospeso restituisce un output simile al seguente:

```
{
    "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"
                }
            ]
        }
    ]
}
```

Puoi assegnare l'elenco a cluster specifici con`--filters`, nel modulo. `Name=`*`filter-name`*`,Values=`*`resource-id`*`,...` Il filtro accettato `Name` è`db-cluster-id`, che contiene un elenco di identificatori di cluster o ARN.

Example

Per Linux, macOS o Unix:

```
aws docdb describe-pending-maintenance-actions \
```

```
    --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Per Windows:

```
aws docdb describe-pending-maintenance-actions ^
    --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Applica le date

Ogni azione di manutenzione in sospeso prevede fino a tre date di applicazione. Compaiono nell' AWS CLI output di `describe-pending-maintenance-actions` e indicano quando verrà eseguita l'azione. I campi sono `null` destinati alla manutenzione opzionale.

- **`CurrentApplyDate`**—quando è pianificata l'esecuzione dell'azione, adesso o nella successiva finestra di manutenzione. Compilato per le azioni obbligatorie e forzate.
- **`AutoAppliedAfterDate`**—la data dopo la quale inizia l'applicazione automatica durante la finestra di manutenzione del cluster o dell'istanza. Compilato per le azioni richieste.
- **`ForcedApplyDate`**—la scadenza rigida. Dopo questa data, l'azione viene eseguita automaticamente, indipendentemente dalla finestra di manutenzione. Popolato per azioni forzate.

Per posticipare un'azione in sospeso, sposta la finestra di manutenzione a un altro giorno prima. `AutoAppliedAfterDate` Una volta `AutoAppliedAfterDate` completata, l'azione verrà applicata automaticamente durante la successiva finestra di manutenzione. Una volta `ForcedApplyDate` superato, non è possibile alcun ulteriore rinvio. L'esatta finestra di rinvio varia a seconda della patch; le date sono pubblicate nella notifica AHD e nell'output. AWS CLI

Aggiornamenti del motore Amazon DocumentDB

Dopo aver identificato una patch del motore in sospeso, utilizza una delle seguenti procedure per applicarla o pianificarla. È possibile eseguire queste procedure da Console di gestione AWS o da. AWS CLI

Using the Console di gestione AWS

Per gestire un aggiornamento per un cluster

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.

2. Nel pannello di navigazione scegliere Cluster.
3. Seleziona il cluster che desideri aggiornare.
4. Dal menu Azioni, scegli una delle seguenti opzioni:

   - Effettua subito l'upgrade: esegui immediatamente la manutenzione in sospeso.
   - Esegui l'aggiornamento nella finestra successiva: eseguilo durante la successiva finestra di manutenzione del cluster.

   Puoi anche utilizzare Applica ora o Applica alla prossima finestra di manutenzione dalla sezione Manutenzione in sospeso della scheda Manutenzione e backup del cluster (vedi). Visualizzazione delle azioni di manutenzione di Amazon DocumentDB in sospeso

   > **Note**
   >
   > Se non c'è nulla in sospeso, tutte queste opzioni sono inattive.

Using the AWS CLI

Applica un aggiornamento in sospeso con. `apply-pending-maintenance-action`

Parameters

- **`--resource-identifier`**—Amazon DocumentDB Amazon Resource Name (ARN) della risorsa a cui si rivolge l'azione in sospeso.
- **`--apply-action`**—l'azione di manutenzione in sospeso da applicare. `system-update`Da utilizzare per applicare una patch al motore.
- **`--opt-in-type`**—il tipo di richiesta di opt-in o se annullarne una. Valori validi:
  - `immediate`—applica ora. Non può essere annullato una volta inviato.
  - `next-maintenance`—applica durante la prossima finestra di manutenzione della risorsa.
  - `undo-opt-in`—annulla un opt-in esistente`next-maintenance`.

Example

Per Linux, macOS o 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
```

Per 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
```

Disponibilità della lettura durante l'applicazione delle patch

I motori Amazon DocumentDB 5.0 e 8.0 preservano la disponibilità di lettura durante l'applicazione delle patch quando il cluster ha più istanze. Amazon DocumentDB corregge le istanze dei lettori in modo continuativo, in tre gruppi, in modo che i lettori rimanenti continuino a erogare traffico. L'autore non è disponibile per un breve periodo durante l'aggiornamento. Per azzerare i tempi di inattività di lettura, impostate le vostre preferenze di lettura in modo che le letture possano ricadere sullo scrittore, `secondaryPreferred` oppure `primaryPreferred` lavorare, oppure, `secondary` da sole, comportare tempi di inattività di lettura. `primary`

Modalità di preferenza di lettura	Durante l'aggiorn amento dello scrittore	Durante l'aggiorn amento del lettore	Numero minimo di lettori necessario per azzerare i tempi di inattività di lettura
`primary`	Read/write tempi di inattività	Nessun impatto	N/A
`primaryPr eferred`	Annota i tempi di inattività	Nessun impatto	1
`secondary`	Annota i tempi di inattività	Tempo di inattività della lettura (se c'è un solo lettore)	2

Modalità di preferenza di lettura	Durante l'aggiorn amento dello scrittore	Durante l'aggiorn amento del lettore	Numero minimo di lettori necessario per azzerare i tempi di inattività di lettura
`secondary Preferred`	Annota i tempi di inattività	Nessun impatto	1
`nearest`	Annota i tempi di inattività	Nessun impatto	1

Mentre i lettori stanno applicando le patch, la velocità di lettura complessiva del cluster diminuisce temporaneamente. Per mantenere costante il throughput, fornisci lettori aggiuntivi prima dell'aggiornamento e rimuovili una volta completato.

Sui motori 3.6 e 4.0, queste funzionalità di disponibilità della lettura non sono applicabili: una patch al motore causa tempi di inattività più lunghi che influiscono sia sulle letture che sulle scritture. Per eseguire l'aggiornamento a una versione principale che lo fa, consulta. Aggiornamento della versione principale in-place di Amazon DocumentDB

Durata dei tempi di inattività delle patch

Engine-patch i tempi di inattività variano. I fattori principali sono l'utilizzo della CPU e la pressione della memoria sull'istanza al momento della patch, quindi è importante dimensionare correttamente le istanze. Per ridurre al minimo i tempi di inattività, esegui l'ultima versione principale del motore di Amazon DocumentDB e distribuisci le istanze su più zone di disponibilità.

Aggiornamenti e sostituzioni delle patch

Amazon DocumentDB monitora le patch dopo il rilascio. Nel raro caso in cui venga identificato un problema, Amazon DocumentDB sospende l'implementazione mentre prepara una versione aggiornata. Quando ciò accade, i cluster che non hanno ancora ricevuto la patch non la vedono più come un'azione di manutenzione disponibile e la corrispondente notifica di modifica pianificata in essa viene ritirata. Health Dashboard I cluster che già eseguono la versione interessata continuano a funzionare normalmente e non richiedono alcuna azione da parte dell'utente.

A breve seguirà una patch aggiornata. Quando sarà disponibile nella tua regione, riceverai una nuova notifica tramite e-mail, come descritto inNotifiche per le patch del motore Amazon DocumentDB. Health Dashboard

Aggiornamenti a versioni secondarie

Amazon DocumentDB pubblica versioni secondarie in aggiunta alla versione principale 5.0 e successive (ad esempio`5.0.1`). Le versioni secondarie non vengono pubblicate per le versioni principali precedenti alla 5.0. Le versioni secondarie si comportano diversamente dalle patch del motore obbligatorie e opzionali:

- Non appaiono come un'azione di manutenzione in sospeso e non si applicano mai automaticamente.
- Non generano notifiche AHD o e-mail. Le nuove versioni secondarie sono annunciate nelle note di rilascio di Amazon DocumentDB.
- Per eseguire l'aggiornamento, modifichi la versione del motore del cluster (immediatamente o durante la successiva finestra di manutenzione). Gli aggiornamenti delle versioni minori richiedono tempi di inattività brevi e sono unidirezionali: non è possibile effettuare il downgrade a una versione secondaria precedente. Per i cluster globali, aggiorna i cluster secondari prima di quelli primari.

Per saperne di più:. Aggiornamento della versione secondaria di Amazon DocumentDB

Aggiornamenti del sistema operativo Amazon DocumentDB

Le istanze necessitano occasionalmente di aggiornamenti del sistema operativo. Amazon DocumentDB aggiorna il sistema operativo per migliorare le prestazioni e rafforzare la sicurezza. Gli aggiornamenti del sistema operativo lasciano invariate la versione del motore del cluster e la classe di istanza. Analogamente alle patch del motore, gli aggiornamenti del sistema operativo utilizzano il ciclo di vita facoltativo/richiesto/forzato descritto all'inizio di questo argomento; a differenza delle patch del motore, un aggiornamento del sistema operativo può passare da una categoria all'altra nel tempo se viene posticipato. Applica gli aggiornamenti del sistema operativo non appena sono disponibili e imposta le finestre di manutenzione del cluster e delle istanze in base agli orari adatti alle esigenze aziendali.

Utilizza l'azione di `os-upgrade` manutenzione a livello di cluster per applicare gli aggiornamenti del sistema operativo a tutte le istanze di un cluster. Amazon DocumentDB aggiorna le istanze in modo continuativo, alcune alla volta, e aggiorna l'istanza principale per ultima per ridurre al minimo i failover. L'aggiornamento viene eseguito durante la finestra di manutenzione del cluster, non la finestra di manutenzione delle singole istanze, che configuri.

Dopo che un'istanza riceve un aggiornamento del sistema operativo, la cache del buffer inizia a essere vuota. Fino a quando il working set non viene ripopolato dal volume di archiviazione, le query su quell'istanza possono avere una latenza maggiore e una minore. `BufferCacheHitRatio`

Quando Amazon DocumentDB aggiorna l'istanza primaria, un failover promuove una replica come nuova istanza primaria. Usa l'endpoint del cluster in modo che la tua applicazione lo gestisca in modo trasparente. Per mantenere la disponibilità di lettura durante l'aggiornamento delle istanze, imposta la preferenza di lettura su `secondaryPreferred` o `primaryPreferred` in modo che le letture possano tornare a un'istanza disponibile. Mantieni le potenziali destinazioni di failover (repliche con il livello di priorità più alto) nella stessa classe di istanza dell'istanza primaria. Ciò evita il peggioramento delle prestazioni di scrittura dopo la promozione. Per informazioni dettagliate, vedi Failover di Amazon DocumentDB.

Le azioni a livello di cluster `os-upgrade` e a livello di istanza potrebbero essere visualizzate contemporaneamente tra le azioni disponibili`system-update`. `describe-pending-maintenance-actions` Tuttavia, non è possibile pianificarle entrambe contemporaneamente. Se `system-update` le azioni a livello di istanza sono pianificate attivamente su qualsiasi istanza, è necessario annullarle o completarle prima di pianificare l'azione a livello di cluster `os-upgrade` e viceversa.

Important

La tua istanza Amazon DocumentDB va offline per l'aggiornamento del sistema operativo. Multi-instance i cluster riducono al minimo l'impatto. Se esegui un cluster a istanza singola, puoi aggiungere temporaneamente un cluster secondario per l'aggiornamento e rimuoverlo in seguito. Il secondario è soggetto ai normali costi finché esiste.

Note

L'`system-update`azione a livello di istanza è ancora disponibile per la compatibilità con le versioni precedenti. Se è necessario utilizzarla, aggiorna prima le repliche e per ultima quella principale: evita di applicarle contemporaneamente, poiché un failover durante la patch può prolungare i tempi di inattività.

Per ricevere un evento quando arriva un nuovo aggiornamento opzionale del sistema operativo, iscriviti alla categoria degli eventi relativi alle patch di sicurezzaRDS-EVENT-0230. Per ulteriori informazioni, consulta Iscrizione agli eventi di Amazon DocumentDB.

> **Note**
>
> Rimanere aggiornati sugli aggiornamenti facoltativi e obbligatori potrebbe essere necessario per garantire la conformità. Applica os-upgrade le azioni regolarmente durante le finestre di manutenzione.

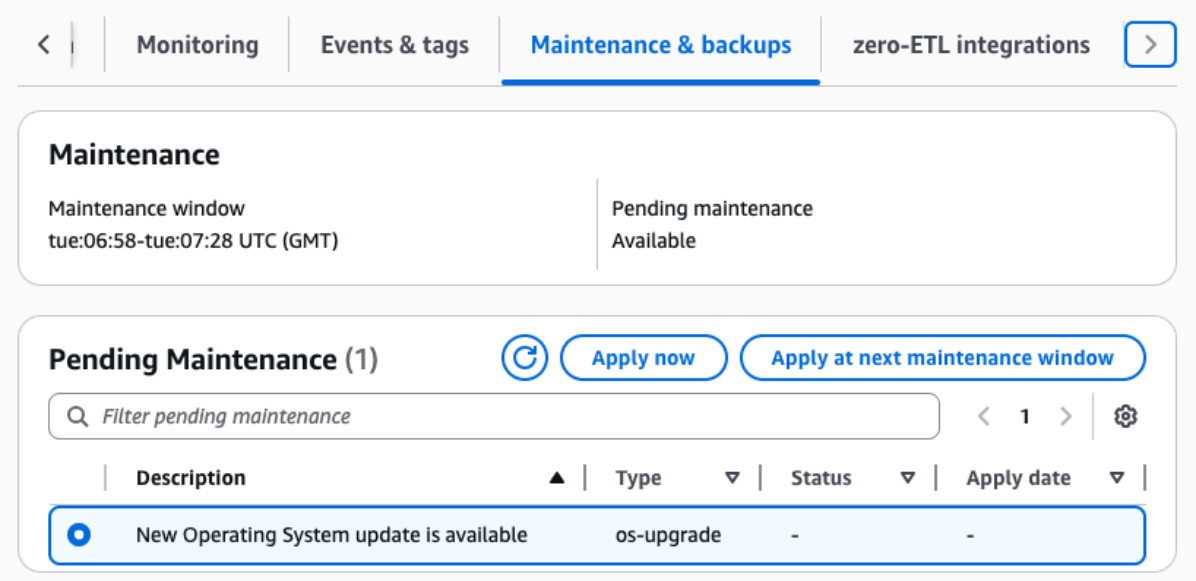
Gli aggiornamenti del sistema operativo sono legati a classi di istanze specifiche, pertanto istanze diverse diventano idonee in momenti diversi. Se sul cluster non è installata l'ultima patch del motore, l'aggiornamento del sistema operativo potrebbe non essere visualizzato: applica prima la patch del motore più recente (vedi). Aggiornamenti del motore Amazon DocumentDB

Usa Console di gestione AWS o AWS CLI per verificare se è disponibile un aggiornamento.

Using the Console di gestione AWS

Per verificare la disponibilità di un aggiornamento del sistema operativo dalla console:

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.
2. Nel pannello di navigazione, scegli Clusters, quindi seleziona il nome del cluster.
3. Scegli la scheda Manutenzione e backup.
4. In Manutenzione in sospeso, l'os-upgradeazione viene visualizzata se è disponibile un aggiornamento del sistema operativo.



5. Seleziona l'`os-upgrade`azione e scegli Applica ora o Applica alla prossima finestra di manutenzione. Se il valore è la finestra successiva, puoi posticipare l'aggiornamento con Refer upgrade fino a quando l'azione non è iniziata.

Using the AWS CLI

Verifica la presenza di un aggiornamento del sistema operativo in sospeso:

```
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"
                }
            ]
        }
    ]
}
```

Gli aggiornamenti del sistema operativo vengono visualizzati a livello di cluster as `os-upgrade` e a livello di istanza as`system-update`. Usa l'azione a livello di cluster`os-upgrade`.

Example

L'esempio seguente applica immediatamente l'aggiornamento del sistema operativo.

Per Linux, macOS o 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
```

Per 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 aggiornamenti

Alcune modifiche vengono eseguite autonomamente, ad esempio sostituendo una classe di istanza con una con più o meno memoria o modificando il gruppo di parametri del cluster. Amazon DocumentDB li tratta in modo diverso dagli aggiornamenti che avvia. Per maggiori dettagli, consulta:

- Modifica di un cluster Amazon DocumentDB
- Modifica di un'istanza Amazon DocumentDB

Per elencare le modifiche avviate dall'utente che sono ancora in sospeso:

Example

Per elencare le modifiche in sospeso avviate dall'utente per le tue istanze

Per Linux, macOS o Unix:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

Per Windows:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

In questo esempio, `sample-cluster-instance` ha una modifica in sospeso in; non ne ha. `db.r5.xlarge` `sample-cluster-instance-2`

```
[
    [
        "sample-cluster",
        "sample-cluster-instance",
        {
            "DBInstanceClass": "db.r5.xlarge"
        }
```

```
    ],
    [
        "sample-cluster",
        "sample-cluster-instance-2",
        {}
    ]
]
```

Applicazione di patch ai cluster globali

In un cluster globale, ogni cluster membro, primario e secondario, si aggiorna durante la propria finestra di manutenzione. Quando una patch del motore richiesta è disponibile in ogni regione, riceverai una notifica AHD e via e-mail. Le patch opzionali e le nuove versioni secondarie non generano notifiche; consulta le note di rilascio di Amazon DocumentDB per scoprirle.

Se ti applichi da solo, applica sempre le patch secondarie per prime e le primarie per ultime. Questo ordine mantiene il failover e lo switchover disponibili per tutta la durata del rollout.

> **Important**
>
> Se correggi per errore la prima versione principale, ripristina tutte le versioni secondarie alla stessa versione il prima possibile. Il failover e lo switchover rimangono disattivati finché tutti i cluster non utilizzano la stessa versione.

Se non intraprendi alcuna azione, la patch si applica automaticamente durante la successiva finestra di manutenzione di ciascun cluster: prima i secondari, poi quelli primari nella relativa finestra una volta completati i secondari.

Mantieni i cluster DB primario e secondario sulla stessa versione. Il failover interregionale gestito funziona solo su un database globale quando ogni cluster condivide la stessa versione del motore e lo stesso livello di patch. Lo stesso vale se aggiungi un nuovo secondario che utilizza una versione del motore più recente rispetto a quella principale: crea nuovi secondari nella versione del primario prima di unirli al database globale.

Dopo la notifica di una patch, aggiorna la versione principale e secondaria alla versione più recente non appena possibile per far funzionare il failover e lo switchover. Se una richiesta di failover o switchover viene rifiutata, confronta le versioni delle patch del motore tra i cluster; se non corrispondono, applica la patch disponibile sui cluster in ritardo.

Comprendere i ruoli collegati ai servizi

Amazon DocumentDB (con compatibilità con MongoDB) utilizza ruoli collegati ai servizi AWS Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo unico di ruolo IAM collegato direttamente ad Amazon DocumentDB. Service-linked i ruoli sono predefiniti da Amazon DocumentDB e includono tutte le autorizzazioni richieste dal servizio per chiamare altri servizi per tuo conto. AWS

Un ruolo collegato ai servizi semplifica l'utilizzo di Amazon DocumentDB perché non è necessario aggiungere manualmente le autorizzazioni necessarie. Amazon DocumentDB definisce le autorizzazioni dei suoi ruoli collegati ai servizi e, salvo diversa definizione, solo Amazon DocumentDB può assumerne i ruoli. Le autorizzazioni definite includono la policy di trust e la policy delle autorizzazioni. Una policy delle autorizzazioni specifica non può essere collegata a un'altra entità IAM.

È possibile eliminare i ruoli solo dopo aver eliminato le risorse correlate. Questo protegge le tue risorse Amazon DocumentDB perché non puoi rimuovere inavvertitamente l'autorizzazione di accesso alle risorse.

Per informazioni su altri servizi che supportano i ruoli collegati ai servizi, consulta AWS Services That Work with IAM e cerca i servizi che hanno Sì nella colonna Ruolo. Service-Linked Scegliere Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato al servizio per tale servizio.

Autorizzazioni per i ruoli collegati ai servizi Amazon DocumentDB

Amazon DocumentDB (con compatibilità con MongoDB) utilizza il ruolo collegato ai servizi denominato AWSServiceRoleForRDS per consentire ad Amazon DocumentDB di chiamare AWS i servizi per conto dei tuoi cluster.

Il ruolo AWSServiceRoleForRDS collegato ai servizi prevede che i seguenti servizi assumano il ruolo:

- `docdb.amazonaws.com`

La politica delle autorizzazioni dei ruoli consente ad Amazon DocumentDB di completare le seguenti azioni sulle risorse specificate:

- Operazioni su `ec2`:

  - `AssignPrivateIpAddresses`
  - `AuthorizeSecurityGroupIngress`
  - `CreateNetworkInterface`
  - `CreateSecurityGroup`
  - `DeleteNetworkInterface`
  - `DeleteSecurityGroup`
  - `DescribeAvailabilityZones`
  - `DescribeInternetGateways`
  - `DescribeSecurityGroups`
  - `DescribeSubnets`
  - `DescribeVpcAttribute`
  - `DescribeVpcs`
  - `ModifyNetworkInterfaceAttribute`
  - `RevokeSecurityGroupIngress`
  - `UnassignPrivateIpAddresses`
- Operazioni su `sns`:
  - `ListTopic`
  - `Publish`
- Operazioni su `cloudwatch`:
  - `PutMetricData`
  - `GetMetricData`
  - `CreateLogStream`
  - `PullLogEvents`
  - `DescribeLogStreams`
  - `CreateLogGroup`

> **Note**
>
> Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato al servizio è necessario configurare le relative autorizzazioni. Potrebbe essere visualizzato il messaggio di errore seguente:

Unable to create the resource. (Impossibile creare la risorsa. Verify that you have permission to create service linked role. (Verifica di possedere le autorizzazioni necessarie per creare un ruolo collegato ai servizi.) Otherwise wait and try again later. (In caso contrario, attendi e riprova più tardi.
Se viene visualizzato questo errore, verifica che le autorizzazioni seguenti siano abilitate:

```
{
    "Action": "iam:CreateServiceLinkedRole",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"rds.amazonaws.com"
        }
    }
}
```

Per ulteriori informazioni, consulta Service-Linked Role Permissions nella IAM User Guide.

Creazione di un ruolo collegato al servizio Amazon DocumentDB

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando crei un cluster, Amazon DocumentDB crea il ruolo collegato al servizio per te.

Se devi ricreare un ruolo collegato ai servizi che hai precedentemente eliminato, puoi utilizzare lo stesso processo per ricreare il ruolo nel tuo account. Quando crei un cluster, Amazon DocumentDB crea nuovamente il ruolo collegato al servizio per te.

Modifica di un ruolo collegato al servizio Amazon DocumentDB

Amazon DocumentDB non consente di modificare il ruolo collegato al servizio. AWSServiceRoleForRDS Dopo avere creato un ruolo collegato al servizio, non sarà possibile modificarne il nome perché varie entità potrebbero farvi riferimento. Tuttavia, puoi modificare la descrizione del ruolo utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un Service-Linked ruolo nella IAM User Guide.

Eliminazione di un ruolo collegato al servizio Amazon DocumentDB

Se non hai più bisogno di utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, elimina quel ruolo. In questo modo non hai un'entità inutilizzata che non viene monitorata o gestita attivamente. Tuttavia, prima di poter eliminare il ruolo collegato ai servizi, dovrai eliminare tutti i cluster .

Pulizia di un ruolo collegato al servizio Amazon DocumentDB

Prima di utilizzare IAM per eliminare un ruolo collegato ai servizi, devi innanzitutto verificare che il ruolo non abbia sessioni attive ed eliminare tutte le risorse utilizzate dal ruolo.

Per verificare la presenza di una sessione attiva del ruolo collegato ai servizi utilizzando la console

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo. https://console.aws.amazon.com/iam/
2. Nel pannello di navigazione della console IAM, scegli Ruoli, quindi scegli il nome (non la casella di controllo) del AWSServiceRoleForRDS ruolo.
3. Nella pagina Summary (Riepilogo) per il ruolo selezionato, scegliere la scheda Access Advisor (Consulente accessi).
4. Nella scheda Access Advisor (Consulente accessi) esaminare l'attività recente per il ruolo collegato ai servizi.

> **Note**
>
> Se non sei sicuro che Amazon DocumentDB stia utilizzando il AWSServiceRoleForRDS ruolo, puoi provare a eliminarlo. Se il servizio sta utilizzando il ruolo, l'eliminazione non andrà a buon fine e potrai visualizzare le regioni in cui il ruolo viene utilizzato. Se il ruolo è in uso, prima di poterlo eliminare dovrai attendere il termine della sessione. Non puoi revocare la sessione per un ruolo collegato al servizio.

Se desideri rimuovere il AWSServiceRoleForRDS ruolo, devi prima eliminare tutte le istanze e i cluster. Per informazioni sull'eliminazione delle istanze e dei cluster, consulta i seguenti argomenti:

- Eliminazione di un'istanza Amazon DocumentDB
- Eliminazione di un cluster Amazon DocumentDB

Regioni supportate per i ruoli collegati al servizio Amazon DocumentDB

Amazon DocumentDB supporta l'utilizzo di ruoli collegati ai servizi in tutte le regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta https://docs.aws.amazon.com/documentdb/latest/devguide/regions-and-azs.html#regions-and-azs-availability.

Ottimizzazione delle prestazioni

Questo capitolo fornisce indicazioni sull'ottimizzazione delle prestazioni di Amazon DocumentDB e sulla risoluzione dei problemi di prestazioni più comuni.

Argomenti

- Problemi di prestazioni comuni
- Anti-patterns
- Suggerimenti per il miglioramento delle prestazioni
- Analisi del piano di interrogazione
- Index-only scansioni (domande coperte)
- Pianifica i filtri della cache
- Le migliori pratiche per Amazon DocumentDB
- Gestione degli indici Amazon DocumentDB
- Gestione della compressione dei documenti a livello di raccolta
- Gestione della compressione basata su dizionari in Amazon DocumentDB 8.0

Problemi di prestazioni comuni

Questa sezione descrive i problemi di prestazioni più comuni che potresti riscontrare con Amazon DocumentDB e come risolverli.

Argomenti

- Elevato utilizzo della CPU
- Problemi di connessione con Amazon DocumentDB
- Interrogazione lenta

Elevato utilizzo della CPU

L'elevato utilizzo della CPU è uno dei problemi che potresti incontrare a un certo punto mentre lavori con Amazon DocumentDB. Questa sezione fornisce approfondimenti per risolvere il problema.

Identificazione: individua il problema

L'elevato utilizzo della CPU sulle istanze di Amazon DocumentDB rallenta le applicazioni aumentando le latenze in tutte le operazioni del database. Quando l'utilizzo della CPU di Amazon DocumentDB supera i livelli ottimali, i tempi di esecuzione delle query aumentano, la creazione della connessione richiede più tempo e la reattività delle applicazioni diminuisce. I picchi della CPU sull'istanza principale rallentano le operazioni di scrittura, mentre i picchi sulle repliche determinano una maggiore latenza per le operazioni di lettura.

Alcune delle cause più comuni di utilizzo elevato della CPU in Amazon DocumentDB potrebbero essere:

- Scansioni delle raccolte: le interrogazioni senza indici appropriati forzano le scansioni complete della raccolta, consumando una quantità eccessiva di CPU per esaminare ogni documento.
- Picchi di connessione: picchi improvvisi nelle connessioni delle applicazioni sovraccaricano le risorse della CPU necessarie per l'autenticazione e la gestione delle sessioni.
- Aggregazioni complesse: le Multi-stage operazioni (ordinamento, raggruppamento, unione) consumano una notevole quantità di CPU, specialmente su set di dati di grandi dimensioni senza indici ottimizzati.
- Interrogazioni inefficienti: l'elevato utilizzo della CPU deriva dalla scansione di set di risultati di grandi dimensioni e dalle operazioni regex.
- Write-Heavy Carichi di lavoro: gli aggiornamenti frequenti e gli inserimenti in blocco mettono a dura prova la CPU durante l'elaborazione dei documenti, gli aggiornamenti degli indici e la registrazione delle transazioni.

Indagate: raccogliete le metriche

Amazon DocumentDB fornisce il monitoraggio tramite Amazon. CloudWatch I parametri di Amazon DocumentDB possono essere ampiamente suddivisi in metriche a livello di cluster e a livello di istanza. Per le risorse relative a CPU, memoria e connessioni, consulta i parametri a livello di istanza poiché il consumo di risorse è specifico dell'istanza. Identifica le istanze (primarie o di replica) che mostrano i picchi della CPU e annota i tempi di picco.

La CloudWatch metrica per l'utilizzo della CPU per. `CPUUtilization`

Diagnosi: trova la causa principale

Dopo aver identificato l'istanza e la tempistica del picco della CPU, si procede alla diagnosi della causa che influisce in generale sull'utilizzo della CPU. Poiché i motivi che possono causare un picco della CPU sono molteplici, cerchiamo di identificare il motivo che vi sta contribuendo, a partire da:

Connessioni: controllate le CloudWatch metriche, `DatabaseConnections` tra cui: se ci sono picchi nelle connessioni e questo coincide con il periodo di picco della CPU, è probabile che l'aumento delle connessioni nel breve periodo sia la causa del picco della CPU. `DatabaseConnectionsMax`

Interrogazioni: se non è la connessione, la causa potrebbe essere una query. Se hai abilitato Performance Insights, vai alla console Performance Insights dell'istanza e visualizza le query sull'istanza. L'altra cosa da guardare alle query è quella di Profiler.

Risolvi: risolvi il problema

Interruzioni di connessione: se le interruzioni di connessione sono la causa di picchi di CPU, valuta la possibilità di riutilizzare le connessioni nell'applicazione o di implementare il pool di connessioni. Per ulteriori informazioni sull'ottimizzazione delle connessioni elevate, consulta. the section called “Problemi di connessione”

Distribuzione delle query: se i picchi della CPU sono causati da query di lettura sull'istanza principale, prendete in considerazione la possibilità di reindirizzare le operazioni di lettura alle istanze di replica se i risultati delle query possono essere alla fine coerenti.

Ottimizzazioni delle query: se una query o un insieme di query causano picchi di CPU, valuta la possibilità di ottimizzare la query, ad esempio aggiungendo un indice se la query non ne utilizza uno. Per ulteriori informazioni sull'ottimizzazione di una query, vedere. the section called “Query in esecuzione lenta”

Scalabilità delle istanze: se l'utilizzo della CPU rimane costantemente elevato dopo gli sforzi di ottimizzazione, esegui l'upgrade a una classe di istanze più ampia con maggiore capacità della CPU per gestire efficacemente i requisiti del carico di lavoro.

Senza server: per carichi di lavoro imprevedibili che causano picchi di CPU di breve durata, prendi in considerazione l'utilizzo. Utilizzo di Amazon DocumentDB serverless Le istanze serverless ridimensionano automaticamente le risorse in base alla domanda, eliminando la necessità di fornire istanze di dimensioni fisse per tali carichi di lavoro.

Problemi di connessione con Amazon DocumentDB

Identificazione: individua il problema

Cause comuni

I problemi di connessione derivano in genere da tre aree principali:

L'esaurimento del pool di connessioni si verifica quando un'applicazione raggiunge il numero massimo di connessioni consentite ad Amazon DocumentDB, attraverso i limiti del pool di connessioni lato client o i limiti delle istanze lato server. Questa condizione comporta un peggioramento delle prestazioni dell'applicazione, timeout e potenziali errori in quanto le nuove richieste di connessione vengono messe in coda o rifiutate.

Il sovraccarico dell'autenticazione si verifica quando Amazon DocumentDB riceve un numero eccessivo di richieste di autenticazione simultanee, in particolare quando si elaborano più di 1.000 nuove connessioni in un breve periodo di tempo. Durante la manutenzione dell'autenticazione, Amazon DocumentDB mantiene un blocco esclusivo sulla mappa delle sessioni, facendo sì che i successivi tentativi di autenticazione restino in coda fino al completamento della manutenzione.

I problemi di configurazione in Amazon DocumentDB spesso derivano da configurazioni errate nelle impostazioni di rete, sicurezza e client. Ciò include elementi quali impostazioni errate dei gruppi di sicurezza, configurazione VPC errata o problemi relativi ai certificati. SSL/TLS Comprendere la corretta configurazione è essenziale per mantenere un accesso sicuro e affidabile al database.

Diagnosi: individuazione della causa principale

Pool di connessione

Il pool di connessioni viene inizializzato durante la creazione di un' MongoClient istanza. Ogni pool mantiene le connessioni in base a due parametri chiave:

min PoolSize - Numero minimo di connessioni mantenute

max PoolSize - Numero massimo di connessioni consentite

Quando una richiesta richiede una connessione:

1. Il pool verifica le connessioni inattive disponibili
2. Se non ne esiste nessuna e la dimensione del pool è inferiore a quella massimaPoolSize, crea una nuova connessione

3. Se è al massimoPoolSize, la richiesta entra in una coda di attesa
4. Se la coda è piena o il timeout è scaduto, viene generata MongoWaitQueueFullException

Il comportamento della coda di attesa viene gestito tramite questi parametri:

wait QueueTimeout MS - Tempo massimo di attesa per la connessione

wait QueueSize - Numero massimo di richieste in coda

Ecco un esempio di approccio problematico per la connessione ad Amazon DocumentDB in cui viene creato un nuovo pool ogni volta:

```
for(Request request : requests) {
    MongoClient client = MongoClients.create(settings);
    // Process request
    client.close();
}
```

CloudWatch Le metriche critiche da monitorare sono:

- `DatabaseConnections`- Il numero di connessioni (attive e inattive) aperte su un'istanza eseguita con una frequenza di 1 minuto.
- `DatabaseConnectionsMax`- Il numero massimo di connessioni al database aperte (attive e inattive) su un'istanza in un periodo di 1 minuto.
- `DatabaseConnectionsLimit`- Il numero massimo di connessioni simultanee al database (attive e inattive) consentite su un'istanza in un dato momento.
- `LowMemNumOperationsThrottled`- Il numero di richieste che vengono limitate a causa della scarsa memoria disponibile in un periodo di 1 minuto.

Per le quote per classe di istanza, vedere. Quote delle istanze

I segnali di allarme più comuni relativi ai problemi del pool di connessioni a livello di applicazione includono:

- Aumento dei tempi di acquisizione delle connessioni
- Aumento delle dimensioni della coda di attesa

- Numero crescente di eccezioni di timeout

Sovraccarico di autenticazione

La connessione ad Amazon DocumentDB segue un flusso simile al seguente:

Richiesta di connessione → Handshake SSL → Autenticazione → Creazione della sessione → Connessione pronta

Quando si elaborano più di 1.000 nuove connessioni, le richieste di connessione aggiuntive entreranno in coda per l'autenticazione dopo aver completato l'handshake SSL. I tempi medi di connessione dell'applicazione aumenteranno durante questi eventi di sovraccarico.

CloudWatch Le metriche critiche da monitorare sono:

- `DatabaseConnections`- Il numero di connessioni (attive e inattive) aperte su un'istanza eseguita con una frequenza di 1 minuto.
- `DatabaseConnectionsMax`- Il numero massimo di connessioni al database aperte (attive e inattive) su un'istanza in un periodo di 1 minuto.
- `DatabaseConnectionsLimit`- Il numero massimo di connessioni simultanee al database (attive e inattive) consentite su un'istanza in un dato momento.

Problemi di configurazione

Il problema di configurazione più comune si verifica quando si tenta di connettersi a un endpoint privato del cluster Amazon DocumentDB da un ambiente senza accesso all'ambiente di rete privato. Amazon DocumentDB è solo un cloud privato virtuale (VPC) e attualmente non supporta endpoint pubblici. Non puoi connetterti direttamente al tuo cluster Amazon DocumentDB dal tuo laptop o da un ambiente di sviluppo locale esterno al tuo VPC.

Ciò si manifesterà in errori come i seguenti:

```
Error: couldn't connect to server...
Failed to connect to...
exception: connect failed
connection attempt failed
```

Anche configurazioni errate dei gruppi di sicurezza possono causare errori di connessione. Per impostazione predefinita, un cluster Amazon DocumentDB ascolta le connessioni sulla porta TCP 27017. L'applicazione fallirà se si tenta di connettersi a una porta diversa da quella con cui è stato distribuito il cluster o se l'applicazione non è coperta dalla configurazione del gruppo di sicurezza di ingresso per il cluster.

Una gestione errata dei certificati può inoltre causare problemi di connessione. Per impostazione predefinita, la crittografia in transito è abilitata per i cluster Amazon DocumentDB appena creati. Quando la crittografia in transito è abilitata, sono necessarie connessioni sicure tramite TLS per connettersi al cluster utilizzando il certificato global-bundle.pem. Se tenti di utilizzare il certificato errato, riceverai errori come:

```
unable to get local issuer certificate
```

Se tenti di connetterti a un cluster con TLS abilitato senza specificare i parametri TLS, riceverai errori come:

```
Server selection timed out after 30000 ms
```

Risolvi: risolve il problema

Pool di connessioni: rivedi il pool di connessioni implementando o regolando le dimensioni del pool in base ai requisiti del carico di lavoro. Le configurazioni ottimali del pool dipendono dal carico di lavoro e dai requisiti. È necessario mantenere un valore minimo PoolSize tale che le connessioni principali siano pronte e disponibili e un valore massimo sufficientemente WaitTime breve da fallire rapidamente se il pool è esaurito.

Ecco un esempio di come riutilizzare un singolo pool senza crearne uno nuovo ogni volta:

```
MongoClient client = MongoClients.create(settings);
    for(Request request : requests) {
    // Process request
}
```

Sovraccarico dell'autenticazione: gestisci l'autenticazione implementando un aumento graduale delle connessioni e limitando le nuove connessioni a 1.000 alla volta. Usa il pool di connessioni per

riutilizzare efficacemente le connessioni autenticate. Per evitare di sovraccaricare il cluster Amazon DocumentDB di connessioni, implementa una strategia di aumento delle connessioni.

```
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;
        }
    }
}
```

Puoi anche configurare le impostazioni del pool di connessioni per limitare il numero totale di connessioni consentite.

```
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();
```

Problemi di configurazione: assicurati che l'applicazione abbia accesso al VPC privato e alla sottorete in cui si trovano le risorse Amazon DocumentDB. Se utilizzi VPC Peering, consulta la guida per sviluppatori Troubleshoot a VPC peering connection per ulteriori informazioni. Puoi anche consultare l'articolo del Knowledge Center Come posso risolvere i problemi di connettività da Internet alle istanze Amazon EC2 all'interno del mio VPC? .

Per la configurazione del gruppo di sicurezza, devi includere una regola di ingresso nel tuo gruppo di sicurezza Amazon DocumentDB per consentire le connessioni dalla tua applicazione.

```
{
  "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"
    }
  ]
}
```

Se il cluster è configurato con crittografia TLS, scarica il certificato TLS per Amazon DocumentDB denominato global-bundle.pem e utilizzalo per la connessione al cluster.

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

Long-term Soluzioni

La scalabilità delle istanze può essere necessaria tramite l'aggiornamento a una classe di istanze più ampia o l'aggiunta di repliche di lettura per distribuire il carico di connessione. Una corretta implementazione del bilanciamento del carico garantisce un utilizzo ottimale delle risorse in tutto il cluster.

Le modifiche alle applicazioni dovrebbero concentrarsi sull'implementazione di una solida gestione delle connessioni, un monitoraggio completo e il rispetto delle migliori pratiche di pool di connessioni. Ciò include una corretta gestione degli errori e una gestione del ciclo di vita delle connessioni.

I miglioramenti dell'architettura potrebbero comportare l'adozione di Amazon DocumentDB Serverless per carichi di lavoro variabili, l'implementazione di una sofisticata logica di riprova e la progettazione in base alla tolleranza agli errori. Prendi in considerazione la possibilità di ristrutturare l'architettura delle applicazioni per gestire meglio la gestione delle connessioni.

Best practice

Pool di connessioni

Attraverso una corretta gestione e monitoraggio dei pool di connessioni, le applicazioni possono mantenere una connettività stabile del database prevenendo scenari di esaurimento che potrebbero influire sull'affidabilità e sulle prestazioni del sistema. Configura i timeout appropriati e ridimensiona il pool in base alle caratteristiche del carico di lavoro.

Esempio di impostazione del pool di connessioni

```
MongoClientSettings settings = MongoClientSettings.builder()
    .applyToConnectionPoolSettings(builder ->
        builder.maxSize(10))
    .applyToConnectionPoolSettings(builder ->
        builder.maxWaitQueueSize(2))
    .applyToConnectionPoolSettings(builder ->
        builder.maxConnectionIdleTime(10, TimeUnit.MINUTES))
    .build();
```

Per ulteriori informazioni, consulta Creazione di applicazioni resilienti con Amazon DocumentDB — Parte 1: Configurazione del client.

Sovraccarico di autenticazione

Implementa sempre il pool di connessioni con valori appropriati per i parametri in base al carico di lavoro. Utilizza una tecnica di creazione graduale delle connessioni e mantieni connessioni permanenti, ove possibile. Implementa una corretta pulizia della connessione per garantire che non vengano sprecate risorse inattive.

Problemi di configurazione

Assicurati di aver configurato il routing appropriato dalla tua applicazione alle risorse Amazon DocumentDB. Utilizza TLS per la crittografia in transito e implementa l'accesso con privilegi minimi. Verifica le tue credenziali Amazon DocumentDB e convalida i valori delle stringhe di connessione.

Interrogazione lenta

Identificazione: individua il problema

Una query a esecuzione lenta è quella che supera il tempo di esecuzione previsto per i requisiti aziendali. La definizione di ciò che costituisce una query lenta varia in base ai diversi carichi di lavoro. È possibile identificare le query lente tramite i log del profiler o Performance Insights.

1. Abilita il profiler se non è già abilitato. I log di Profiler vengono scritti su Amazon nel CloudWatch gruppo di log:. `/aws/docdb/<cluster-identifier>/profiler` Usa CloudWatch Logs Insights per interrogarli.

   Esempio di query di analisi dei CloudWatch log per ottenere le 10 query più lente per la raccolta ecommerce.products:

   ```
   filter ns="ecommerce.products" | sort millis desc | limit 10
   ```

2. Usa Performance Insights per identificare query costose quasi in tempo reale. Abilita Performance Insights se non è già abilitato.

   a. Apri la AWS console, accedi ad Amazon Amazon DocumentDB, quindi a Performance Insights e seleziona l'istanza del cluster.

   b. Consulta la cronologia del caricamento del database (AAS) e le query più frequenti (in base al carico del database). Espandi un digest di query per visualizzare le dichiarazioni secondarie letterali.

   c. Acquisisci le domande che richiedono un'analisi.

   > **Note**
   >
   > Non tutte le query in Performance Insights possono essere inefficienti o lente.

Indagare: raccogli informazioni

1. Il profiler fornisce il piano di esecuzione delle query e le metriche chiave ad esso associate, tra cui millis, nreturn e planSummary (utilizzo dell'indice):

```
{
    "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"
}
```

Per trovare le query COLLSCAN:

```
filter planSummary="COLLSCAN" | sort millis desc | limit 20
```

2. Usa Performance Insights per analizzare le tendenze di esecuzione delle query, come attese, carico e impatto sulle risorse (ad esempio, IOPS o CPU per query) in tempo reale.

   Poiché Performance Insights non fornisce il piano di esecuzione delle query, acquisisci la query ed esegui explain («ExecutionStats») sulla query sul tuo cluster Amazon DocumentDB:

```
db.ecommerce.products.find().explain("executionStats")
```

3. Facoltativamente, correla le metriche del profiler con i dati di Performance Insights (ad esempio, abbina le query high millis in profiler con le query principali in Performance Insights).

Diagnosi: individua la causa principale

In questo passaggio, diagnostica il piano di interrogazione per identificare potenziali ottimizzazioni. Utilizza il seguente flusso: sintomo → causa probabile:

- Sintomo: PlanSummary: «COLLSCAN»

  Causa: indice mancante o errato.
- Sintomo: aggregazione lenta con $group/ $sort

  Causa: la pipeline di aggregazione potrebbe elaborare troppi dati in memoria.
- Sintomo: latenze elevate mentre Performance Insights mostra il carico del database raggruppato per attese IO, sebbene vengano utilizzati indici (PlanSummary: «IXSCAN»).

  Causa: le query I/O associate; gli indici o il set di lavoro superano la cache del buffer disponibile sull'istanza.
- Sintomo: PI mostra gli stati di attesa della CPU, un AAS elevato a causa del numero limitato di query

Causa: CPU-bound interrogazioni (aggregazioni complesse, $regex).

- Sintomo: molte query lente ma nessuna mostra un costoso PlanSummary

  Causa: collo di bottiglia esterno (rete, limitazione, attività di manutenzione, volume di interrogazioni).

Risolvi: risolvi il problema

Quando implementi le correzioni, esegui sempre la convalida con explain («ExecutionStats») e monitora il carico del database di Performance Insights.

1. Riepilogo del piano: «COLLSCAN»
   - Crea un indice mirato.

     Esempio: per le query frequenti che filtrano per {category, price} e ordinano per prezzo decrescente:

     ```
     db.products.createIndex({ category: 1, price: -1 })
     ```

2. aggregazione lenta con $group/$sort
   - Premi $match e $project in anticipo per ridurre il flusso di documenti in $group /$sort.
   - Limita il numero di campi nelle prime fasi della 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. latenze elevate mentre Performance Insights mostra DB Load raggruppato per attese IO
   - Verifica se BufferCacheHitRatio è basso o se i ReadiOps sull'istanza sono alti. Aumenta la memoria dell'istanza (aggiorna la classe dell'istanza, ad esempio r6g.large → r6g.xlarge) o usa la classe di istanza NVMe.

- Riduci l'ingombro dell'indice.
- Aggiungi repliche di lettura per scaricare il traffico di lettura (usa l'impostazione ReadPreference per reindirizzare il traffico sulle repliche se la query è tollerante con eventuale coerenza).

4. PI mostra gli stati di attesa della CPU, un AAS elevato a causa di poche query
   - Sostituisci i costosi $regex con ricerche con prefisso indicizzato o indice $text.
   - Scritture in batch (inserimento) per ridurre l'amplificazione della scrittura.

> **Note**
>
> Verifica sempre le modifiche in un ambiente inferiore (non di produzione) prima di promuovere tali modifiche a Production.

Anti-patterns

Questa sezione illustra gli anti-pattern più comuni che possono influire negativamente sulle prestazioni di Amazon DocumentDB.

Argomenti

- Indice composto con più di 3 attributi
- Interrogazioni di lunga durata
- Indici poco utilizzati e ridondanti
- Multi-Key Indici con matrici di grandi dimensioni

Indice composto con più di 3 attributi

Panoramica di

Un indice composto o composito mantiene i riferimenti a più campi all'interno di un'unica struttura di indice. Questi indici ottimizzano le prestazioni per le query che filtrano su più campi contemporaneamente o combinano il filtraggio con operazioni di ordinamento. Sono efficaci anche per le interrogazioni a condizione singola sui campi indicizzati più a sinistra. Il database sfrutta queste voci di indice per individuare in modo efficiente i documenti corrispondenti senza eseguire scansioni complete della raccolta.

Amazon DocumentDB può utilizzare l'indice composto per supportare query che includono qualsiasi sottoinsieme principale dei campi indicizzati, un concetto noto come prefissi di indice. Ad esempio, se hai un indice composto attivo`{state: 1, city: 1, zipcode: 1}`, Amazon DocumentDB può elaborare in modo efficiente le query che utilizzano solo il campo 'state', i campi 'state' e 'city' insieme o tutti e tre i campi 'state', 'city' e 'zipcode'. Tuttavia, le query devono utilizzare i campi da sinistra a destra senza saltare i campi intermedi. Ciò significa che le query che utilizzano solo il campo «città» o «codice postale» o combinazioni come «stato» e «codice postale» (saltando «città»), non possono utilizzare appieno l'indice.

Nella maggior parte degli scenari del mondo reale, gli indici composti con tre o meno attributi in genere garantiscono prestazioni ottimali ed efficienza delle risorse. Sebbene sia possibile avere più di 3 attributi nell'indice composito, spesso porta a un maggiore consumo di risorse, che supera i vantaggi.

Impatto sul cluster

Sebbene si sia tentati di creare un indice che copra tutte le condizioni di una query per ottenere le massime prestazioni teoriche, la maggior parte del filtraggio dei dati avviene tramite i primi 1-3 attributi di un indice composito. I campi aggiuntivi che superano questa soglia contribuiscono principalmente alla dimensione dell'indice piuttosto che a un'ottimizzazione significativa delle query.

Archiviazione e I/O costi generali: gli indici compositi con molti attributi consumano molto più spazio di archiviazione rispetto alle alternative più semplici. La dimensione è direttamente proporzionale al numero di attributi indicizzati e alla dimensione dei valori indicizzati stessi.

Impronta di memoria: l'ampio spazio di archiviazione degli indici compositi con molti attributi crea un'impronta corrispondente in memoria, ampliando il set di lavoro e sostituendo altri dati a cui si accede di frequente dal pool di buffer.

Operazioni di scrittura: ogni modifica del documento che interessa più campi indicizzati richiede l'aggiornamento dell'intera voce dell'indice composito, moltiplicando il lavoro necessario per completare l'operazione di scrittura.

Come identificare

Inizia esaminando tutti gli indici della tua raccolta per identificare gli indici compositi con più di tre attributi:

```
// 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 }
```

Correzione

Se viene utilizzato un indice composito con più di tre attributi, identifica le query che lo utilizzano e cerca opportunità di ottimizzazione. Valuta la possibilità di sostituire questi indici con altri più efficienti contenenti tre o meno attributi. Dopo aver implementato i nuovi indici, rimuovete i vecchi indici che contengono più di tre attributi.

> **Note**
>
> Coordinatevi sempre con le parti interessate e convalidate l'impatto sulle prestazioni prima di eliminare qualsiasi indice.

Segui la regola Equality, Sort, Range (ESR) durante la creazione di indici compositi.

Regola ESR (Equality Sort Range): questo ordinamento massimizza l'efficienza dell'indice filtrando prima il set di dati con condizioni di uguaglianza, quindi applicando operazioni di ordinamento sul set ridotto e infine eseguendo scansioni degli intervalli sul sottoinsieme più piccolo possibile

- ordina i campi come Uguaglianza (corrispondenze esatte),
- Ordinamento (ordinamento),
- Intervallo (>, <, $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
```

Interrogazioni di lunga durata

Panoramica di

Long-running le query possono causare problemi di prestazioni a livello di cluster. Queste query possono interferire con il processo di raccolta dei rifiuti Multi-Version Concurrency Control (MVCC) di Amazon DocumentDB, con conseguente accumulo di documenti con versioni precedenti e un peggioramento delle prestazioni in tutto il cluster. Amazon DocumentDB implementa un timeout lato server di 2 ore come meccanismo di sicurezza per impedire che le query inutilizzate consumino risorse a tempo indeterminato.

Cosa sono le query a lunga durata:

- Query eseguite per periodi prolungati (in genere > 30 minuti)
- Apri i cursori che rimangono attivi per ore (il timeout di 2 ore non è applicabile se il cursore è attivo)

Impatto sul cluster

Le interrogazioni di lunga durata possono interferire con il processo di raccolta dei rifiuti

- Le versioni precedenti si accumulano: Garbage collector non può recuperare le vecchie versioni dei documenti
- Incremento della raccolta e dell'indice: le voci relative alla raccolta e all'indice si accumulano nel tempo, aumentano e ciò può comportare un aumento dei costi di archiviazione.
- Pressione della CPU e della memoria: la pressione sulla CPU e sulla memoria aumenta a causa dell'elaborazione inefficiente di un maggior numero di vecchie versioni di documenti, voci di indice e ID di transazione.

Query a esecuzione prolungata → Blocchi GC → Storage Growth → Pressione della CPU e della memoria → Altre query lunghe

Monitora e rileva

1. Per trovare query a esecuzione prolungata, utilizzare il comando CurrentOp.

```
// 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. Per trovare cursori attivi per più di 30 minuti

```
// 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. Monitoraggio dei progressi di Garbage Collector tramite CloudWatch

`LongestRunningGCProcess`— Durata in secondi del processo di raccolta dei rifiuti attivo più lungo. Si aggiorna ogni minuto e tiene traccia solo delle operazioni attive, esclusi i processi che vengono completati entro la finestra di un minuto.

`AvailableMVCCIds`-Un contatore che mostra il numero di operazioni di scrittura rimanenti disponibili prima di raggiungere lo zero. Quando questo contatore raggiunge lo zero, il cluster entra in modalità di sola lettura fino a quando gli ID non vengono recuperati e riciclati. Il contatore diminuisce a ogni operazione di scrittura e aumenta man mano che la Garbage Collection ricicla i vecchi ID MVCC.

> **Note**
>
> Gli ID MVCC inferiori e la durata estesa della Garbage Collection non sono attribuiti esclusivamente a query di lunga durata. Write-intensive i carichi di lavoro su istanze con risorse limitate possono inoltre comportare una riduzione della disponibilità degli ID MVCC e prolungare i cicli di raccolta dei rifiuti.

Strategie di riparazione

- Implementa i timeout delle query nell'applicazione
- Non mantenete attivi i cursori per periodi prolungati
- Ottimizza le query per prestazioni migliori.
- Preferisci il raggruppamento in batch delle operazioni di scrittura

Indici poco utilizzati e ridondanti

Panoramica di

Gli indici migliorano le prestazioni complessive delle query dei carichi di lavoro, ma comportano dei costi. Ogni indice consuma spazio di archiviazione e I/O risorse e le operazioni di scrittura devono aggiornare tutti gli indici della raccolta. Se il database contiene indici inutilizzati, sottoutilizzati o ridondanti, ciò comporta un sovraccarico per il cluster, con conseguente peggioramento delle prestazioni. La presenza di più indici non necessari crea un anti-pattern che causa problemi di prestazioni, aumento dei costi di storage e maggiore sovraccarico operativo.

Scenari di indicizzazione non ottimali

- Indici non utilizzati: indici creati per interrogazioni occasionali o in iterazioni precedenti del prodotto a cui non si accede più dai modelli di query correnti. Per identificare gli indici non utilizzati in una raccolta, puoi utilizzare IndexStats. Per ulteriori informazioni, consulta the section called “Analizza l'utilizzo degli indici e identifica gli indici non utilizzati”.
- Indici ridondanti: più indici che coprono modelli chiave sovrapposti o modelli di query identici. Esempio: un indice a chiave singola sull'attributo A diventa ridondante quando un indice composito copre sia l'attributo A che B, poiché l'indice composito può gestire le query solo sull'attributo A.
- Over-Indexing: creazione di indici «per ogni evenienza» senza analizzare i modelli di query o i requisiti prestazionali effettivi.

- Indici a bassa cardinalità: indici su campi con pochi valori distinti (ad esempio campi booleani, indicatori di stato) che offrono vantaggi minimi di ottimizzazione delle query.

Impatto sulle prestazioni del cluster

- Storage e IO: ogni indice consuma spazio di storage e risorse IO, contribuendo ai costi complessivi. Se un indice è inutilizzato o ridondante, rappresenta un costo non necessario per il cluster.
- Prestazioni di scrittura ridotte: le operazioni di inserimento, aggiornamento ed eliminazione devono mantenere gli indici, generando un sovraccarico che consuma risorse. Gli indici non utilizzati o ridondanti aggiungono un sovraccarico non necessario a queste operazioni.
- Memoria e pressione della CPU: gli indici competono per la memoria del pool di buffer e possono eliminare i dati a cui si accede di frequente se il workset non rientra nella memoria dell'istanza, con conseguenti risultati non ottimali. BufferCacheHitRatio Il sistema consuma anche risorse della CPU per il mantenimento delle strutture degli indici.

Strumenti disponibili per identificare le opportunità di ottimizzazione degli indici

- Strumento di revisione dell'indice:
- Strumento di rilevamento della cardinalità dell'indice

Raccolta del toolkit:

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

1. Strumento di revisione dell'indice

Lo strumento di revisione degli indici analizza tutte le raccolte e gli indici per fornire un elenco di indici, i relativi modelli di utilizzo e identifica gli indici ridondanti nel cluster Amazon DocumentDB. Esegui questo strumento su tutte le istanze del cluster per un'analisi completa. Per ulteriori informazioni, consulta Index Review Tool.

Utilizzo:

```
cd performance/index-review/
```

```
python3 index-review.py --server-alias <server-alias> --uri <mongodb-uri>
```

File di output

- `{server-alias}-collections.csv`: Collection-level statistiche
- `{server-alias}-indexes.csv`: metriche dettagliate sull'utilizzo dell'indice
- `{server-alias}-{timestamp}-index-review.json`: Dati grezzi per ulteriori analisi

> **Note**
>
> Esegui per ogni istanza del cluster con un alias indipendente per generare statistiche da ogni istanza, quindi esegui lo strumento per ottenere un'analisi cumulativa

Ad esempio: `python3 index-review.py --files "{server-alias}-{timestamp}-index-review.json, {server-alias}-{timestamp}-index-review.json" --server-alias full-review`

2. Strumento di rilevamento della cardinalità dell'indice

Questo strumento identifica gli indici con bassa cardinalità che sono inefficienti per le prestazioni delle query.

Utilizzo

```
cd performance/index-cardinality-detection/

python3 detect-cardinality.py --uri <mongodb-uri>
```

Parametri

- `--threshold`: percentuale della soglia di cardinalità (impostazione predefinita: 1%)
- `--sample-count`: Numero di documenti da campionare (impostazione predefinita: 100.000)
- `--max-collections`: numero massimo di raccolte da scansionare per database (impostazione predefinita: 100)

Per ulteriori informazioni, consulta https://aws.amazon.com/blogs/database/detect-and-fix-low-cardinality-indexes-in-amazon-documentdb/

Strategie di riparazione

1. Rimozione degli indici inutilizzati e ridondanti

```
// Example: Drop unused index
db.collection.dropIndex("unused_index_name")
```

> **Note**
>
> Gli indici non devono mai essere eliminati senza averne discusso con tutte le parti interessate e aver testato le prestazioni.

2. Ottimizzazione dell'indice di bassa cardinalità

- Usa indici parziali con filtri
- Converti singoli indici a bassa cardinalità in indici composti

Multi-Key Indici con matrici di grandi dimensioni

Panoramica di

Multi-key gli indici in Amazon DocumentDB consentono di interrogare in modo efficiente i campi dell'array. Questa strategia di indicizzazione offre latenze di query notevolmente inferiori durante la ricerca all'interno dei dati dell'array, rendendola un'opzione interessante per applicazioni con strutture di dati complesse.

Tuttavia, quando si lavora con array di grandi dimensioni o più campi di array all'interno di una raccolta, è importante comprendere le implicazioni in termini di risorse e le caratteristiche prestazionali per prendere decisioni architettoniche informate. Quando crei un indice su un campo array, Amazon DocumentDB genera voci di indice individuali per ogni elemento dell'array.

Impatto sul cluster

- Storage e sovraccarico di I/O: a volte Multi-key gli indici possono occupare spazio di archiviazione fino a più volte rispetto alla tabella di base, le dimensioni sono direttamente proporzionali ai documenti della raccolta con l'attributo array indicizzato, al numero di elementi nell'array e alla dimensione dell'elemento.
- Utilizzo della memoria: poiché l'ingombro di archiviazione è relativamente ampio, questi indici hanno un'impronta corrispondente in memoria, il che si traduce in un set di lavoro più ampio.
- Operazioni di scrittura: gli Multi-key indici creano un sovraccarico durante le operazioni di scrittura. Ogni elemento dell'array genera voci di indice separate, moltiplicando il lavoro richiesto per completare l'operazione di scrittura.

Strategie di riparazione

- Crea un indice a più chiavi se necessario e limita il numero di campi nell'array.
- Limita il numero di indici a più chiavi nella raccolta
- Prendi in considerazione la possibilità di modificare il tuo modello di dati

Suggerimenti per il miglioramento delle prestazioni

Questa sezione fornisce cinque strategie di ottimizzazione delle prestazioni per Amazon DocumentDB per migliorare l'efficienza delle applicazioni e l'esecuzione delle query.

1. Usa $match come prima fase nelle pipeline di aggregazione

Posiziona sempre $match come prima fase di filtraggio nella pipeline di aggregazione per massimizzare le prestazioni. Amazon DocumentDB utilizzerà gli indici in modo efficace quando $match guiderà la pipeline, permettendo al database di filtrare i dati in anticipo e ridurre il sovraccarico di elaborazione.

```
// Optimized approach
db.orders.aggregate([
  { $match: { status: "active", category: "electronics" } }, // Index utilization
  { $group: { _id: "$category", total: { $sum: "$price" } } },
  { $sort: { total: -1 } }
```

```
])
```

Impatto: il filtraggio precoce riduce il numero di documenti elaborati nelle fasi successive della pipeline, con conseguente esecuzione più rapida delle query e un minore consumo di risorse.

2. Usa $project in Aggregation Pipeline per ridurre al minimo le dimensioni dei dati della pipeline

Trasporta solo i campi essenziali nelle fasi della pipeline di aggregazione per ridurre al minimo le dimensioni dei dati e migliorare le prestazioni. Usa $project in modo strategico per includere solo i dati di cui hai bisogno.

```
// 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" } } }
])
```

Impatto: documenti più piccoli riducono l'utilizzo della memoria e migliorano l'efficienza di elaborazione della pipeline, con conseguente miglioramento delle prestazioni complessive delle query.

3. Abilita la compressione dei documenti per ridurre i costi e i costi di archiviazione e migliorare le prestazioni delle query I/O

Abilita la compressione dei documenti dal gruppo di parametri del cluster per ridurre i costi e i I/O costi di archiviazione e migliorare le prestazioni delle query. Amazon DocumentDB archivia i documenti compressi su disco e nella RAM, riducendo l'ingombro e i costi della memoria. I/O

Impatto:

- La memoria disponibile contiene più documenti
- Accesso più rapido ai dati con letture su disco ridotte
- Riduzione dei costi e dei I/O costi di storage e prestazioni di query migliorate

> **Note**
>
> Amazon DocumentDB non abilita la compressione per impostazione predefinita per la versione 5.0. Puoi abilitare la compressione a livello di raccolta o di cluster per il cluster 5.0. Utilizza l'utilità di revisione della compressione di Amazon DocumentDB per analizzare i rapporti di compressione delle tue raccolte.
> Per Amazon DocumentDB 8.0, la compressione è abilitata per impostazione predefinita.

4. Sfrutta gli indici per prestazioni di query ottimali

Assicurati che le tue query utilizzino sempre gli indici per prestazioni ottimali. Amazon DocumentDB offre diversi tipi di indice per soddisfare diversi casi d'uso.

Principi di indicizzazione:

- Ogni query deve utilizzare un indice appropriato
- Amazon DocumentDB fornisce diversi tipi di indice
- Gli indici composti offrono la massima flessibilità in quanto supportano varie forme di interrogazione con un unico indice
- Progetta indici per supportare contemporaneamente le operazioni di ordinamento e filtraggio

Comprensione dei prefissi degli indici: gli indici composti funzionano tramite i prefissi di indice: Amazon DocumentDB può utilizzare qualsiasi sottoinsieme da sinistra a destra dei campi indice. Ad esempio, `{ category: 1, price: -1, inStock: 1 }` l'indice crea i seguenti prefissi utilizzabili:

- `{ category: 1 }`- supporta il filtraggio delle query solo per categoria
- `{ category: 1, price: -1 }`- supporta il filtraggio delle query per categoria e per prezzo sorting/filtering
- `{ category: 1, price: -1, inStock: 1 }`- supporta la query composta completa

Le query su price, InStock o InStock da sole non utilizzeranno questo indice poiché non iniziano con il primo campo (categoria).

Come identificare le query che non utilizzano indici: utilizzate il metodo explain () per analizzare l'esecuzione delle query e identificare le query che eseguono scansioni di raccolta anziché utilizzare indici.

Impatto: le query senza utilizzo dell'indice generano scansioni della raccolta, con conseguente aumento della pressione della memoria e della CPU sull'istanza e una maggiore latenza delle query.

5. Ottimizza i modelli di dati in base ai modelli di query

Allinea il tuo modello di dati al modo in cui l'applicazione interroga e aggiorna i dati. La modellazione dei dati è alla base delle applicazioni Amazon DocumentDB ad alte prestazioni.

Strategie di ottimizzazione:

Incorporamento per le prestazioni

- Archivia insieme i dati correlati quando vi si accede frequentemente come unità
- Incorpora documenti che vengono sempre recuperati insieme
- Adatto per relazioni uno-a-pochi

```
// 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" }
  ]
}
```

Riferimento per la flessibilità

- Utilizza i riferimenti per dati di grandi dimensioni o a cui si accede raramente
- Consigliato per relazioni uno-a-molti con set di dati di grandi dimensioni

- Previene il sovraccarico dei documenti e migliora le prestazioni di aggiornamento

Strategia di suddivisione della raccolta

Quando solo pochi campi di documenti di grandi dimensioni vengono aggiornati frequentemente, o quando dati di grandi dimensioni a cui si accede raramente sovraccaricano i documenti, prendi in considerazione la possibilità di suddividere le raccolte:

- Conserva i campi aggiornati di frequente in una raccolta separata e più piccola
- Archivia i dati statici o a cui si accede raramente in un'altra raccolta
- Collegali ai riferimenti quando necessario

```
// 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 */]
}
```

Aumento delle prestazioni: documenti più piccoli significano aggiornamenti più rapidi, utilizzo ridotto della memoria e maggiore efficienza della cache.

Impatto: una modellazione inefficiente dei dati comporta interrogazioni non ottimali, maggiori dimensioni dei documenti e un utilizzo elevato della memoria, con conseguente peggioramento delle prestazioni delle applicazioni e aumento dei costi operativi.

Analisi del piano di interrogazione

L'analisi del piano di query tramite explain plan fornisce informazioni essenziali sulle prestazioni delle query di Amazon DocumentDB. Il piano di query con ExecutionStats rivela metriche chiave, tra cui:

- Documenti restituiti per fase (nReturned)
- Stage-specific tempi di esecuzione (esecuzioneTimeMillisEstimate)
- Durata della generazione del piano (pianificazioneTimeMillis)

Esaminando l'output del piano di query, gli sviluppatori possono analizzare i modelli di esecuzione, valutare l'utilizzo degli indici e identificare potenziali opportunità di ottimizzazione nelle fasi della pipeline di query.

Per analizzare il piano di interrogazione, è possibile utilizzare il comando explain () nei seguenti formati.

```
db.runCommand({explain: {query document}, verbosity: "executionStats"})
db.collection.find().explain("executionStats");
```

Di seguito è riportato un esempio di operazione:

```
db.collection.find({ companyname: { '$eq': 'ANYCOMPANY' }, isDeleted: { '$eq':
 false } }).sort({"createdAt":1}).limit(2).explain("executionStats");
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
  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 })
}
```

Di seguito è riportata un'analisi dettagliata di un piano di esecuzione delle query di Amazon DocumentDB, che analizza ogni componente e le sue caratteristiche prestazionali.

Tempistica complessiva

l'esecuzione TimeMillis rappresenta il tempo totale impiegato dall'interrogazione, incluso il tempo di pianificazione.

la pianificazione TimeMillis rappresenta il tempo totale di pianificazione impiegato dall'interrogazione.

fasi di esecuzione

Descrive il processo dettagliato utilizzato da Amazon DocumentDB per eseguire una query, mostrando come i dati fluiscono attraverso diverse operazioni.

```
"executionStages": {
    "stage": "[STAGE_NAME]",
    "nReturned": "[NUMBER_OF_DOCS]",
    "executionTimeMillisEstimate": "[TIME]",
```

```
    "inputStage": {
        // Nested stages
    }
}
```

Fasi comuni di un piano di interrogazione

Di seguito sono riportate le fasi di esecuzione più comuni di un piano di interrogazione. Ogni fase restituisce le metriche di esecuzione TimeMillisEstimate (tempo di esecuzione) e nReturned (numero di documenti) per aiutare a valutare le prestazioni delle query in ogni fase.

COLLSCAN (scansione della raccolta)

- Scansiona l'intera collezione documento per documento
- nReturned: restituisce tutti i documenti corrispondenti nella raccolta
- Può essere un'operazione costosa, vista quando non è disponibile alcun indice

IXSCAN (scansione dell'indice)

- Utilizza un indice per trovare i documenti corrispondenti
- nRestituito: solo i documenti corrispondenti in base all'indice
- Funzionamento efficiente, preferito rispetto a COLLSCAN

SORT

- Ordina i documenti in base a campi specifici
- Se l'attributo sort nella query non fa parte di un indice, questa fase verrà visualizzata in modo esplicito
- nRestituito: stesso numero dei documenti di input
- Memory-intensive per set di risultati di grandi dimensioni. Per ottimizzare, aggiungi il nome del campo di ordinamento a un indice

LIMIT_SKIP

- Controlla il numero di documenti restituiti e ignorati
- nRestituito: limitato dal valore LIMIT

- Utilizzato per operazioni di impaginazione o LIMIT

SOTTOSCANSIONE

- Esegue operazioni di interrogazione annidate
- nReturned: varia in base ai risultati delle subquery
- Utilizzato in interrogazioni complesse con più fasi

Gli stadi ad alta esecuzione TimeMillisEstimate sono buoni candidati per l'ottimizzazione.

> **Note**
>
> Il parametro ExecutionStats attualmente non supporta i comandi di aggiornamento ed eliminazione.

Comprensione dei documenti esaminati in Explain Plan Execution Stats

Quando esegui una query con`explain("executionStats")`, Amazon DocumentDB fornisce parametri di esame che ti aiutano a capire quanti documenti sono stati scansionati per produrre i risultati della query. Queste metriche sono utili per identificare piani di query inefficienti e ottimizzare le prestazioni.

> **Note**
>
> Disponibile solo in Amazon DocumentDB 5.0.1+ e 8.0.0+.

Campi

Campo	Description	Livello
totale DocsExamined	Numero totale di documenti esaminati in tutte le fasi di esecuzione.	Top-level Execution Stats
Documenti esaminati	Numero di documenti esaminati in una fase di esecuzione specifica.	Stage-level

Il `docsExamined` campo viene visualizzato nelle seguenti fasi:

Stage	Description
COLLSCAN	Scansione della collezione. Tutti i documenti della collezione vengono esaminati.
È UNA SCANSIONE	Scansione dell'indice. Vengono esaminati solo i documenti che corrispon dono alla condizione dell'indice.
FETCH	Quando l'ottimizzatore recupera i documenti in una fase separata da IXSCAN, la fase FETCH riporta DocExamined. Nei piani di scansione degli indici, gli stadi IXSCAN secondari non riportano DocExamined. Nei piani $lookup, sia la fase FETCH che le relative fasi secondarie possono riportare DOCExamined.

> **Note**
>
> `docsExamined`non viene visualizzato nelle fasi IXONLYSCAN perché la query viene soddisfatta interamente dall'indice senza accedere ai documenti.

Mostra DocExamined nell'output Explain Plan ExecutionStats

Gli esempi seguenti utilizzano una raccolta con 500.000 documenti e dimostrano come `docsExamined` cambia nelle diverse fasi di esecuzione delle query.

Crea dati di esempio:

```
for (let i = 0; i < 500000; i++) {
    db.coll.insertOne({ number: i, arr: [i, [i+1]], value: "test", bool: i % 2 === 
 0 });
}
```

Scansione della raccolta (COLLSCAN)

Senza un indice, Amazon DocumentDB esegue una scansione completa della raccolta.

```
db.coll.find({ "number": { "$lt": 500 } }).explain("executionStats").executionStats
```

Output:

```
{
    "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
            }
        }
    }
}
```

Tutti i 500.000 documenti sono stati esaminati per restituire 500 risultati. La creazione di un indice nel campo numerico migliora questa situazione.

Scansione dell'indice (IXSCAN)

```
db.coll.createIndex({ number: 1 });
db.coll.find({ "number": { "$lt": 5000 } }).explain("executionStats").executionStats
```

Output:

```
{
    "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
                    }
                }
            ]
        }
    }
}
```

Con l'indice, solo 5.000 documenti su 500.000 sono stati esaminati e recuperati, corrispondenti al numero restituito. La condizione dell'indice ha filtrato 495.000 documenti che non corrispondevano alla query.

Scansione dell'indice con filtro residuo

```
db.coll.find({ "number": { "$lt": 5000 }, "arr": { "$gt":
 4000 } }).explain("executionStats").executionStats
```

Output:

```
{
    "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
            }
        }
    }
}
```

La condizione dell'indice corrispondeva a 5.000 documenti su 500.000, quindi il filtro residuo attivo ha `arr` ridotto il risultato a 999. Il `docsExamined` valore di 5.000 riflette tutti i documenti esaminati dopo la condizione dell'indice ma prima dell'applicazione del filtro residuo.

Effettua il recupero con IXSCAN

```
db.coll.find({ "$or": [{ "number": { "$lt": 100000 } }, { "number": { "$gt":
 400000 } }] }).explain("executionStats").executionStats
```

Output:

```
{
    "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
                                }
                            }
                        ]
                    }
                }
            ]
```

```
        }
    }
}
```

Quando Amazon DocumentDB Optimizer utilizza una fase di recupero per recuperare i documenti, la fase FETCH riporta i risultati. `docsExamined` Le fasi secondarie IXSCAN non generano report `docsExamined` perché scansionano solo le chiavi dell'indice senza accedere direttamente ai documenti.

FETCH con Aggregate Lookup

```
db.coll.explain("executionStats").aggregate([
    { $match: { "number": { "$lt": 5 } } },
    { $lookup: { from: "coll", pipeline: [{ $match: { "number": { "$lt": 3 } } }], as:
 "sub" } }
]).executionStats
```

Output:

```
{
    "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
                                    }
                                }
                            ]
                        }
                    }
                }
            }
        ]
    }
}
```

L'IXSCAN esterno ha esaminato 5 documenti corrispondenti a un numero < 5. L'IXSCAN interno ha esaminato 3 documenti corrispondenti al numero < 3 e la fase FETCH ha esaminato 1 risultato aggregato. Il numero `totalDocsExamined` di 9 è la somma di tutti gli stadi (5 + 3 + 1).

Index-only scansioni (domande coperte)

Una scansione solo indicizzata (chiamata anche query coperta) è un'ottimizzazione in cui Amazon DocumentDB risponde a una query utilizzando interamente i campi memorizzati in un indice, senza leggere i documenti sottostanti. Poiché i valori richiesti da una query sono già presenti nell'indice, Amazon DocumentDB può saltare il recupero di ogni documento corrispondente, evitando il costo casuale I/O e di decodifica della lettura di documenti completi. Nel piano di interrogazione, come `IXONLYSCAN` fase viene visualizzata una scansione solo indicizzata anziché una `IXSCAN` (scansione dell'indice) o (scansione della raccolta). `COLLSCAN` Ciò riduce I/O e può ridurre significativamente la latenza delle query supportate.

Una query è un buon candidato per la scansione del solo indice quando tocca solo un numero limitato di campi che sono tutti presenti in un singolo indice, soprattutto quando i documenti stessi sono di grandi dimensioni. Esempi comuni sono i dashboard, i rollup di analisi e le ricerche che leggono alcuni campi da un indice composto più ampio.

I risultati di una query sono gli stessi indipendentemente dal fatto che venga utilizzata o meno una scansione solo indicizzata. Cambiano solo il piano di interrogazione e la quantità di I/O operazioni eseguite.

> **Note**
>
> Index-only il supporto per la scansione dipende dalla versione del motore Amazon DocumentDB:
>
> - `find`Le proiezioni coperte che includono più di un campo diverso dal `_id` campo richiedono la versione del motore 8.0.2 o successiva. Nelle versioni precedenti, una `find` proiezione coperta può includere al massimo una proiezione non di campo. `_id` Una proiezione con due o più campi ricade su una normale scansione dell'indice o di una raccolta.
> - Index-only le scansioni alla ricerca di pipeline di aggregazione, ad esempio coperte`$group`, richiedono la versione del motore 8.0.2 o successiva.

Quando Amazon DocumentDB utilizza una scansione solo indicizzata

Amazon DocumentDB considera una scansione solo indicizzata per le seguenti operazioni quando un singolo indice contiene tutti i campi letti dall'operazione.

Proiezioni relative ai reperti coperti

Una `find` query si qualifica quando la sua proiezione è una semplice inclusione di uno o più campi di primo livello e ogni campo proiettato è presente in un unico indice. Ad esempio, dato un indice composto attivo`{ customerId: 1, status: 1, total: 1 }`, viene trattata la seguente query. Tutti e tre i campi proiettati sono inclusi nell'indice e `_id` sono esplicitamente esclusi. (La copertura di una proiezione con più di un non `_id` campo richiede la versione del motore 8.0.2 o successiva).

```
db.orders.createIndex({ customerId: 1, status: 1, total: 1 })

db.orders.find(
  { customerId: 42 },
  { _id: 0, customerId: 1, status: 1, total: 1 }
)
```

Il piano vincente utilizza uno `IXONLYSCAN` stage e non recupera mai i documenti corrispondenti dall'archivio:

```
{
  "stage": "IXONLYSCAN",
  "indexName": "customerId_1_status_1_total_1",
  "direction": "forward",
  "indexCond": { "$and": [ { "customerId": { "$eq": 42 } } ] }
}
```

> **Note**
>
> In una `find` query coperta, i campi del risultato vengono restituiti nell'ordine dell'indice (l'ordine in cui i campi appaiono nella definizione dell'indice), non nell'ordine in cui sono stati elencati nella proiezione o nell'ordine in cui sono archiviati nel documento. Questo corrisponde al comportamento di MongoDB per le query coperte. Una query a cui viene data

risposta da una scansione della raccolta restituisce invece i campi nell'ordine dei documenti archiviati.

Aggregazioni di $group coperte

Una `$group` fase in genere legge solo pochi campi (la chiave di gruppo e gli input per i relativi accumulatori) ma spesso viene eseguita senza un'interlinea. `$match` Senza un indice di copertura, Amazon DocumentDB legge e decodifica ogni documento completo della raccolta solo per inserire quei pochi campi nel raggruppamento.

Index-only le scansioni per le pipeline di aggregazione, comprese le `$group` fasi coperte, sono disponibili nella versione del motore 8.0.2 e successive.

Quando un singolo indice contiene tutti i campi letti dal prefisso della pipeline, Amazon DocumentDB può inserire una scansione solo indicizzata sotto il gruppo e leggere quei campi direttamente dall'indice. Considera una pipeline che raggruppa gli ordini per stato e ne somma i totali, con un indice composto su: `{ status: 1, total: 1 }`

```
db.orders.createIndex({ status: 1, total: 1 })

db.orders.aggregate([
  { $group: { _id: "$status", revenue: { $sum: "$total" } } }
])
```

La pipeline legge solo i dati `status` e `total` i dati sono entrambi presenti nell'indice, quindi il gruppo viene alimentato da una scansione `IXONLYSCAN` anziché da una raccolta:

```
// 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)
```

È supportato un valore iniziale `$match` `$sort``$skip`,, o `$limit` precedente. `$group` Tuttavia, ogni campo a cui fanno riferimento queste fasi (ad esempio, un campo `$match` filtro o una `$sort` chiave)

deve essere presente anche nello stesso indice per la copertura della pipeline. Se nell'indice manca un campo di riferimento, Amazon DocumentDB ricorre a una raccolta o a una scansione dell'indice.

> **Note**
>
> A differenza di una `find` proiezione, a `$group` non include implicitamente il documento. `_id` La `_id` in a `$group` è la chiave di gruppo, non l'identificatore del documento, quindi non è necessario che l'indice contenga il documento `_id` per coprire la pipeline.

Requisiti per una domanda coperta

Affinché Amazon DocumentDB possa utilizzare una scansione solo indicizzata, devono essere soddisfatti tutti i seguenti requisiti:

- Un singolo indice contiene tutti i campi letti dalla query. Ciò include i campi proiettati e la chiave di gruppo, gli input dell'accumulatore e qualsiasi campo di filtro o ordinamento per una pipeline coperta. `find` `$group` Se nell'indice manca anche un solo campo, la query non può essere coperta.
- La proiezione è un'inclusione di campi di primo livello (per`find`). Le proiezioni di esclusione, i campi calcolati e gli operatori di proiezione come e non sono idonei. `$slice` `$elemMatch`
- Il **`_id`** campo è contabilizzato (per). `find` Poiché `_id` è incluso per impostazione predefinita, un indice composto comune come not *`{ a: 1, b: 1 }`* include `find({}, { a: 1, b: 1 })`: l'implicito non `_id` è nell'indice. Escludi `_id` dalla proiezione (`{ _id: 0, a: 1, b: 1 }`) o aggiungi `_id` all'indice.
- Il filtro è completamente soddisfatto dall'indice. Se Amazon DocumentDB non è in grado di trasformare parte del predicato della query in un indice associato, mantiene tale parte come filtro residuo, legge il documento per valutarlo e non converte la scansione in una scansione solo indicizzata. Operatori come`$mod`,`$ne`, e `$exists` lasciano un filtro `$nin` residuo.
- Nessun campo proiettato è una colonna a più chiavi (matrice). Consulta Restrizioni.

Restrizioni

Nei seguenti casi non è possibile utilizzare una scansione solo indice e tornare a una normale scansione dell'indice o a una scansione di raccolta. Questo è un comportamento previsto e non influisce mai sulla correttezza dei risultati.

- Campi multikey (array). Se un campo ha mai contenuto un valore di matrice in un documento, la sua colonna di indice è multichiave. Un indice a più chiavi memorizza una voce per elemento dell'array, quindi l'array originale non può essere ricostruito dall'indice. Qualsiasi interrogazione che legge tale campo non è coperta.
- Percorsi punteggiati o annidati. Una proiezione o un riferimento di campo su un percorso annidato, ad esempio, non `"address.city"` è idoneo, anche se l'indice contiene quel percorso.
- Proiezioni di esclusione, calcolate o effettuate dall'operatore. Le proiezioni come `{ b: 0 }{ total: { $add: ["$price", "$tax"] } }`, o `{ a: { $slice: 5 } }` richiedono il documento completo e non sono coperte.
- Whole-document riferimenti in aggregazione. Un documento `$group` che fa riferimento all'intero documento (ad esempio`{ $push: "$$ROOT" }`) non può essere coperto, perché nessun indice può ricostruire l'intero documento.
- Rimodellare le fasi precedenti. **`$group`** Una fase che rimodella i documenti (come`$project`, `$addFields`$unwind, o`$lookup`) che appaiono prima della `$group` squalifica della pipeline dall'essere coperti.
- Mancata corrispondenza tra le regole di confronto sui predicati di stringa. Se le regole di confronto della query sono diverse dalle regole di confronto dell'indice, i predicati di stringa non possono essere serviti dall'indice e lasciano un filtro residuo, che impone il recupero del documento. I predicati numerici e basati sui tipi sono inalterati.

Conferma di una scansione solo indicizzata

Utilizza l'`explain("executionStats")`output per confermare che una query è servita da una scansione solo indicizzata e che non riguarda i documenti. Cerca due cose:

- Il piano vincente utilizza il `IXONLYSCAN` palcoscenico.
- Il `docsExamined` valore in quella fase è `0` (ed `totalDocsExamined` è`0`), il che significa che alla domanda è stata data risposta interamente dall'indice senza leggere alcun documento.

```
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**
>
> Un `docsExamined` valore diverso da zero su uno `IXONLYSCAN` stage indica che l'indice conteneva i dati, ma Amazon DocumentDB ha comunque controllato lo storage per confermare che ogni versione del documento fosse visibile. Ciò può accadere quando la raccolta contiene scritture recenti le cui versioni obsolete del documento non sono ancora state recuperate. Una volta che Garbage Collection elabora quei documenti, passa a. `docsExamined` `0` A quel punto, la query consente di sfruttare appieno i vantaggi della scansione coperta. Per ulteriori informazioni sulla raccolta dei rifiuti, consulta. Raccolta dei rifiuti in Amazon DocumentDB Per ulteriori informazioni sul parametro `docsExamined`, consulta Analisi del piano di interrogazione.

Best practice

- Progetta indici composti che contengano i campi su cui vengono proiettate o raggruppate le query più frequenti, in modo che tali domande possano essere coperte.
- Escludi `_id` dalle `find` proiezioni (`{ _id: 0, ... }`) a meno che non faccia parte dell'`_id`indice di copertura.
- Evita di indicizzare i campi dell'array (multikey) quando intendi coprire una query, perché le colonne a più chiavi non qualificano il percorso di solo indice.

- Consenti al garbage collection di tenere il passo con il tuo carico di lavoro di scrittura, in modo che le scansioni coperte saltino l'accesso ai documenti e rimangano invariate. `docsExamined 0` Per ulteriori informazioni, consulta Raccolta dei rifiuti in Amazon DocumentDB.

Pianifica i filtri della cache

Un filtro cache del piano, chiamato anche filtro di indice, limita il set di indici che il pianificatore di query è autorizzato a considerare per una specifica forma di query. Quando una query corrisponde a una forma dotata di un filtro, il planner sceglie un piano solo tra gli indici indicati in quel filtro invece di valutare tutti gli indici della raccolta.

Una forma di interrogazione è la combinazione del predicato della query, della specifica di ordinamento e dello spazio dei nomi della raccolta, con i valori del predicato normalizzati. Due query che differiscono solo nei valori condividono una forma, quindi un singolo filtro le copre tutte. Nell'output di`planCacheListFilters`, i valori normalizzati sono mostrati come. `"@"`

Un filtro della cache del piano consente di bloccare il piano, in modo da evitare di scegliere un indice più lento a causa della distribuzione dei dati o delle modifiche all'indice. I filtri hanno le seguenti proprietà:

- Non è richiesta alcuna modifica dell'applicazione. I filtri vengono impostati con un comando del database e applicati sul server, in modo da poter mitigare una regressione senza la distribuzione del codice. Questa è la differenza principale rispetto a `ahint`, che deve essere aggiunta in ogni sito di chiamata.
- L'ambito è una forma di interrogazione. Una forma è definita dalla raccolta, dal predicato e dall'ordinamento, non dalla query completa, quindi un filtro vincola solo le query che corrispondono a quella forma. Altre interrogazioni sulla stessa raccolta continuano a essere pianificate normalmente.
- Il cambiamento è duraturo. I filtri persistono durante i riavvii delle istanze e l'applicazione di patch al motore. Se si elimina la raccolta, i relativi filtri vengono rimossi.
- La modifica è reversibile. La cancellazione del filtro riporta la forma alla normale pianificazione basata sui costi, quindi un filtro è un modo a basso rischio per stabilizzare un carico di lavoro mentre si analizza la causa sottostante.

Argomenti

- Comandi supportati

- Forme di interrogazione supportate
- Impostazione, elenco e cancellazione dei filtri
- Verifica dell'applicazione di un filtro
- Esempi
- Comportamento con suggerimenti e modifiche all'indice

Comandi supportati

I filtri della cache del piano richiedono la versione di Planner 2.0 o successiva. Il comando a cui è possibile applicare un filtro dipende dalla versione del planner e dalla versione secondaria di Amazon DocumentDB, come mostrato nella tabella seguente.

Supporto dei comandi per i filtri della cache del piano

Comando	Versione minima del pianificatore	Versione secondaria minima
`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**
>
> La versione 3.0 di Planner è disponibile solo in Amazon DocumentDB 8.0. Poiché `aggregate` i comandi `distinct` and richiedono la versione 3.0 di Planner, i filtri della cache del piano per questi due comandi sono disponibili solo in Amazon DocumentDB 8.0.

In Amazon DocumentDB 5.0, che supporta la versione 2.0 di Planner, i filtri si applicano ai comandi`find`,,`count`, `update` e. `delete` `findAndModify`

Per il `aggregate` comando, la forma viene presa dalla parte anteriore della pipeline: la parte principale `$match` fornisce il filtro e una `$sort` immediatamente dopo fornisce l'ordinamento. `$skip`e `$limit` non influiscono sulla forma, così come non lo fanno gli stadi successivi a quel punto.

Amazon DocumentDB ottimizza la pipeline prima di pianificarla e un filtro della cache del piano viene confrontato con la pipeline ottimizzata. Il filtro e l'ordinamento visualizzati dal pianificatore possono quindi differire dalle fasi che hai scritto. Ad esempio, due `$match` fasi adiacenti nella parte anteriore di una pipeline vengono combinate in un unico `$and` predicato:

```
db.orders.aggregate([
  { $match: { status: "SHIPPED" } },
  { $match: { region: "us-east-1" } },
  { $group: { _id: "$customerId", total: { $sum: "$amount" } } }
])
```

Il planner ne vede uno `$match` attivo`{$and: [{status: ...}, {region: ...}]}`, quindi il filtro deve essere impostato su quel predicato combinato.

```
db.runCommand({planCacheSetFilter: "orders",
query: { $and: [ { status: "SHIPPED" }, { region: "us-east-1" } ] },
indexes: [ "status_1_region_1" ]})
```

Un filtro impostato sulla raccolta esterna di uno `$graphLookup` stadio `$lookup` o determina l'indice utilizzato per scansionare quella raccolta esterna.

Forme di interrogazione supportate

A partire da Amazon DocumentDB 5.0.2 e 8.0.2, i seguenti operatori possono apparire in una forma di query con un filtro della cache del piano:

- `$text`. La stringa di ricerca è normalizzata, pertanto le interrogazioni che differiscono solo per i termini ricercati condividono una forma.
- `$near` e `$nearSphere`. Entrambi gli operatori vengono normalizzati alla stessa forma. Le coordinate`$minDistance`, e non `$maxDistance` fanno parte della forma.

- `$geoWithin` e `$geoIntersects`. Il tipo di geometria GeoJSON fa parte della forma, quindi una `Polygon` query e un'interrogazione sono forme `MultiPolygon` diverse. Le coordinate non fanno parte della forma.
- `$regex`, se usato direttamente su un campo. Il modello e le opzioni sono normalizzati, quindi tutte le espressioni regolari sullo stesso campo condividono una forma.

L'impostazione di un filtro ha esito negativo e viene generato il codice di errore 303 per una forma di interrogazione che utilizza `$jsonSchema``$expr`, o`$sampleRate`. Inoltre, non funziona per le seguenti forme. Un'interrogazione che ne utilizza una qualsiasi è pianificata senza filtro:

- Un `$regex` annidato all'interno`$in`,`$nin`, o`$all`. Un array che combina espressioni regolari con valori letterali non può essere valutato rispetto a un indice, quindi un filtro su quella forma non potrebbe mai essere applicato.
- Una specifica di ordinamento di. `{$natural: 1}` Un `$natural` ordinamento richiede una scansione della raccolta, il che è incompatibile con la limitazione del planner a un indice.

Impostazione, elenco e cancellazione dei filtri

Per impostare un filtro della cache del piano, usa il `planCacheSetFilter` comando. I `sort` campi `query` e insieme specificano la forma della query ed `indexes` elenca i nomi di indice che il planner può prendere in considerazione per quella forma.

```
db.runCommand({
planCacheSetFilter: <collection>,
query: <query>,
sort: <sort>, // optional
indexes: [ <index1>, <index2>, ...],
comment: <any> // optional
})
```

Ad esempio, il comando seguente limita il planner all'a_1indice della forma `{a: {$eq: "@"}, b: {$eq: "@"}}` senza alcun ordinamento:

```
db.runCommand({planCacheSetFilter: "foo",
query: { a: 1, b: 1 },
sort: {},
indexes: [ "a_1" ]})
```

L'impostazione di un filtro per una forma che ne ha già uno sostituisce il filtro esistente. Dopo aver eseguito i due comandi seguenti, il filtro sulla forma `{a: {$eq: "@"}, b: {$eq: "@"}}` è`["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" ]})
```

Per elencare tutti i filtri di una raccolta, usa il `planCacheListFilters` comando:

```
db.runCommand({planCacheListFilters: <collection>})
```

L'output mostra ogni forma memorizzata con i suoi valori normalizzati e il relativo elenco di indici consentiti:

```
{
"filters" : [
{
"query" : { "a" : { "$eq" : "@" } },
"sort" : { },
"indexes" : [ "a_1" ]
},
{
"query" : { "a" : { "$gt" : "@" } },
"sort" : { "a" : 1 },
"indexes" : [ "a_1_b_1" ]
}
],
"ok" : 1
}
```

L'operatore viene preservato e solo il valore viene sostituito con`"@"`. Un'uguaglianza implicita come `{a: 1}` è elencata nella sua forma esplicita e `{"a": {"$eq": "@"}}` gli elementi di un `$in` array vengono sostituiti nel loro insieme, come. `{"a": {"$in": "@"}}` Un set di filtri senza ordinamento viene elencato con un documento vuoto`sort`.

Per rimuovere i filtri, utilizzare il `planCacheClearFilters` comando.

```
db.runCommand({
planCacheClearFilters: <collection>,
query: <query pattern>, // optional
sort: <sort specification>, // optional
comment: <any> // optional
})
```

Per cancellare tutti i filtri della raccolta, omettili entrambi `query` e`sort`:

```
db.runCommand({planCacheClearFilters: "foo"})
```

Fornendo entrambi`query`, `sort` viene cancellato solo il filtro con quell'esatto ordinamento. I valori in ingresso `query` vengono normalizzati, quindi qualsiasi valore rappresentativo funziona. Per cancellare solo il filtro impostato senza un ordinamento, passate un `sort` documento vuoto:

```
db.runCommand({planCacheClearFilters: "foo", query: {a: 1}, sort: {a: 1}})

db.runCommand({planCacheClearFilters: "foo", query: {a: 1}, sort: {}})
```

Verifica dell'applicazione di un filtro

L'`explain`output include due campi che segnalano lo stato del filtro della cache del piano:

- `indexFilterSet`è `true` quando esiste un filtro nell'insieme per la forma della query.
- `indexFilterApplied`è `true` solo quando l'interrogazione è stata pianificata utilizzando un indice dall'elenco di quel filtro.

A partire da Amazon DocumentDB 5.0.2 e 8.0.2, questi campi vengono segnalati per ogni comando che supporta i filtri della cache del piano e nell'output del profiler relativo a tali comandi. Nelle versioni secondarie precedenti venivano segnalati solo per il comando. `find` Per ulteriori informazioni sulla lettura dell'output di explain, vedereAnalisi del piano di interrogazione.

Un filtro può corrispondere a una forma di interrogazione senza essere applicato. Se nessuno degli indici nell'elenco del filtro è in grado di soddisfare la query, il pianificatore ripianifica senza il filtro e sceglie un indice sui costi. In tal caso è ed è. `indexFilterSet` `true` `indexFilterApplied` `false` Ciò accade quando gli indici denominati non coprono i campi della query, quando gli indici

denominati non esistono o quando una `$regex` forma non può produrre limiti di indice, ad esempio un pattern non ancorato o senza distinzione tra maiuscole e minuscole.

Esempi

Gli esempi seguenti utilizzano una `orders` raccolta, e nell'esempio una raccolta, con questi indici: `customers` `$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
```

Esempio: aggiungere una **`query di`** ricerca a un indice composto

Considerate una query che filtra `status` e ordina in base a: `orderDate`

```
db.orders.find({ status: "SHIPPED" }).sort({ orderDate: 1 })
```

L'indice composto `status_1_orderDate_1` soddisfa sia il predicato che l'ordinamento, quindi non è necessario un ordinamento separato. Se invece è il planner a sceglierestatus_1, la query deve ordinare i risultati, il che comporta un aumento di `SORT` livello e diventa più costoso man mano che aumenta il numero di ordini corrispondenti. Per limitare il planner all'indice composto di questa forma:

```
db.runCommand({planCacheSetFilter: "orders",
query: { status: "SHIPPED" },
sort: { orderDate: 1 },
indexes: [ "status_1_orderDate_1" ]})
```

Poiché i valori sono normalizzati all'esterno della forma, questo filtro copre `{ status: "PENDING" }` anche tutti gli altri valori `status` con lo stesso ordinamento. `{ status: "CANCELLED" }` Non copre lo stesso predicato con un ordinamento diverso o con nessun ordinamento, che sono forme separate. Conferma che il filtro è entrato in vigore con`explain`:

```
db.orders.find({ status: "SHIPPED" }).sort({ orderDate: 1 }).explain()
```

I report di output `indexFilterSet: true` `indexFilterApplied: true` e i nomi delle `IXSCAN` fasi`status_1_orderDate_1`.

Esempio: aggiunta di un **aggiornamento** a un indice selettivo

Un aggiornamento deve individuare i documenti da modificare prima di poterli scrivere e tale ricerca utilizza un indice nello stesso modo in cui lo fa una lettura. Quando un predicato può essere servito da più di un indice, l'indice scelto dal planner determina il numero di documenti esaminati dall'aggiornamento.

```
db.orders.updateMany(
  { customerId: 4815, status: "PENDING" },
  { $set: { status: "CANCELLED" } }
)
```

Entrambi `status_1` possono servire a questo `customerId_1` predicato. Per aggiungere l'aggiornamento a`customerId_1`:

```
db.runCommand({planCacheSetFilter: "orders",
query: { customerId: 4815, status: "PENDING" },
indexes: [ "customerId_1" ]})
```

I filtri sui comandi di scrittura richiedono Amazon DocumentDB 5.0.2 o 8.0.2 e Planner versione 2.0 o successiva. Verifica con uno `explain` degli aggiornamenti:

```
db.runCommand({explain: {update: "orders",
updates: [{ q: { customerId: 4815, status: "PENDING" },
             u: { $set: { status: "CANCELLED" } },
             multi: true }]})
```

Il piano vincente è una `UPDATE` fase dopo l'`IXSCAN`altra`customerId_1`. Lo stesso filtro si applica `delete` ai `findAndModify` comandi che utilizzano questa forma di interrogazione, poiché la forma non include il tipo di scrittura.

Esempio: aggiungere una **pipeline** aggregata e una **raccolta esterna $lookup**

Per un'aggregazione, la forma viene presa dalla `$match` fase principale e subito dopo. `$sort` Le fasi successive, ad esempio `$group` o`$project`, non fanno parte della forma.

```
db.orders.aggregate([
  { $match: { status: "SHIPPED" } },
```

```
  { $group: { _id: "$customerId", total: { $sum: "$amount" } } }
])
```

Un filtro impostato sul `$match` predicato determina l'indice utilizzato per leggere`orders`:

```
db.runCommand({planCacheSetFilter: "orders",
query: { status: "SHIPPED" },
indexes: [ "status_1" ]})
```

I filtri si applicano anche alla raccolta a cui si aggiunge uno `$lookup` o uno `$graphLookup` stage. Il filtro è impostato sulla raccolta esterna, non sulla raccolta su cui viene eseguita la pipeline. Nella pipeline seguente, `orders` è la raccolta esterna ed `customers` è la raccolta esterna:

```
db.orders.aggregate([
  { $lookup: { from: "customers",
               localField: "customerId",
               foreignField: "customerId",
               as: "customer" } }
])
```

Il join esegue una verifica `customers` una volta per documento esterno, quindi l'indice selezionato viene applicato ripetutamente e il relativo costo viene moltiplicato in tutta la pipeline. Per aggiungere la scansione a un indice specifico, imposta il filtro in base alla forma `customers` di uguaglianza della chiave di join:

```
db.runCommand({planCacheSetFilter: "customers",
query: { customerId: 1 },
indexes: [ "customerId_1" ]})
```

Poiché i valori sono normalizzati a partire dalla forma, il valore 1 sopra riportato è un segnaposto e qualsiasi valore produce lo stesso filtro. Quando viene applicato un filtro di raccolta estera, il rapporto relativo `true` al livello superiore `indexFilterSet` e ai `indexFilterApplied` campi e l'indice selezionato dal filtro viene visualizzato sul lato esterno del piano vincente. I filtri `aggregate` richiedono Amazon DocumentDB 8.0.2 e Planner versione 3.0.

Comportamento con suggerimenti e modifiche all'indice

- Un filtro della cache del piano ha la precedenza su un. `hint` Amazon DocumentDB applica prima il filtro e poi applica il suggerimento all'interno dell'elenco degli indici consentiti del filtro. Se il

suggerimento nomina un indice consentito dal filtro, viene utilizzato quell'indice. Se il suggerimento nomina un indice non consentito dal filtro, il suggerimento viene ignorato e il pianificatore sceglie tra gli indici consentiti in base ai costi. Entrambi i moduli di suggerimento si comportano allo stesso modo, sia che tu dia un nome all'indice o ne fornisca il modello chiave.

Ad esempio, se la raccolta `foo` contiene gli indici`a_1`, e `b_1a_1_b_1`, e si imposta un filtro sulla forma `{query: {a: {$eq: "@"}, b: {$eq: "@"}}, sort: {a: 1}}` con l'elenco degli indici, l'esecuzione viene `db.foo.find({ a: 10, b: 20 }).sort({a: 1}).hint({ a: 1 })` utilizzata `["a_1", "a_1_b_1"]a_1`, poiché è denominata nel suggerimento e consentita dal filtro.

- L'eliminazione di un indice non modifica i filtri. Il filtro mantiene il nome dell'indice. L'indice non è presente, ma la pianificazione lo ignora. Se un altro indice nell'elenco del filtro può fornire la query, il filtro viene comunque applicato. Il rapporto viene `indexFilterApplied` generato solo quando nessuno degli indici denominati è disponibile. `false` La ricreazione di un indice con lo stesso nome lo rende nuovamente idoneo.
- L'eliminazione di una raccolta rimuove tutti i filtri della raccolta.

Le migliori pratiche per Amazon DocumentDB

Scopri le best practice per lavorare con Amazon DocumentDB (con compatibilità MongoDB). Questa sezione viene continuamente aggiornata man mano che vengono identificate nuove best practice.

Argomenti

- Linee guida operative di base
- Dimensionamento delle istanze
- Utilizzo degli indici
- Best practice di sicurezza
- Ottimizzazione dei costi
- Utilizzo di parametri per identificare problemi a livello di prestazioni
- Carichi di lavoro TTL e serie temporali
- Migrazioni
- Lavorare con i gruppi di parametri del cluster
- Interrogazioni sulla pipeline di aggregazione
- BatchInsert e BatchUpdate

Linee guida operative di base

Le seguenti sono linee guida operative di base che tutti dovrebbero seguire quando lavorano con Amazon DocumentDB. L'accordo sul livello di servizio di Amazon DocumentDB richiede il rispetto di queste linee guida.

- Distribuisci un cluster composto da due o più istanze Amazon DocumentDB in due zone di disponibilità. AWS Per i carichi di lavoro di produzione, distribuisci un cluster composto da tre o più istanze Amazon DocumentDB in tre zone di disponibilità.
- Utilizza il servizio nei limiti indicati. Per ulteriori informazioni, consulta Quote Amazon DocumentDB.
- Monitora l'utilizzo della memoria, della CPU, delle connessioni e dello storage. Per aiutarti a mantenere le prestazioni e la disponibilità del sistema, configura Amazon in CloudWatch modo da notificarti quando i modelli di utilizzo cambiano o quando ti avvicini alla capacità della tua distribuzione.
- Incrementa la capacità delle istanze quando stai per raggiungere i limiti della capacità di storage. È necessario eseguire il provisioning delle istanze con risorse di calcolo sufficienti (ad esempio, RAM, CPU) per soddisfare aumenti imprevisti della domanda da parte delle applicazioni.
- Imposta il periodo di retention dei backup per allineare l'obiettivo del punto di ripristino.
- Prova il failover per il cluster per capire quanto tempo impiega il processo per il tuo caso d'uso. Per ulteriori informazioni, consulta Failover di Amazon DocumentDB.
- Connettiti al tuo cluster Amazon DocumentDB con l'endpoint del cluster (vediEndpoint Amazon DocumentDB) e in modalità replica set (vediConnessione ad Amazon DocumentDB come set di repliche) per ridurre al minimo l'impatto di un failover sulla tua applicazione.
- Scegli un'impostazione di preferenza di lettura del driver che massimizza il dimensionamento della lettura nel rispetto dei requisiti di coerenza di lettura dell'applicazione. La preferenza di lettura `secondaryPreferred` consente la lettura delle repliche e libera l'istanza primaria per eseguire ulteriori operazioni. Per ulteriori informazioni, consulta Leggi le opzioni di preferenza.
- Progetta la tua applicazione per essere resiliente in caso di errori della rete e del database. Usa il meccanismo di errore del driver per distinguere tra errori temporanei ed errori persistenti. Ripeti gli errori temporanei utilizzando un meccanismo di backoff esponenziale quando appropriato. Assicurati che l'applicazione consideri la coerenza dei dati quando implementa una logica di ripetizione.
- Attiva la protezione dell'eliminazione dei cluster per tutti i cluster di produzione o per qualsiasi cluster contenente dati importanti. Prima di eliminare un cluster Amazon DocumentDB, scatta un'istantanea finale. Se stai distribuendo risorse con CloudFormation, abilita la protezione dalla

terminazione. Per ulteriori informazioni, consulta Protezione dalla terminazione e protezione dall'eliminazione.

- Quando si crea un cluster Amazon DocumentDB, `--engine-version` è un parametro opzionale il cui valore predefinito è l'ultima versione principale del motore. L'attuale versione predefinita del motore è 5.0.0 (nota: Amazon DocumentDB 8.0 è disponibile ma deve essere specificata esplicitamente). Quando vengono rilasciate nuove versioni principali del motore, la versione predefinita del motore `--engine-version` verrà aggiornata per riflettere l'ultima versione principale del motore. Di conseguenza, per i carichi di lavoro di produzione, e in particolare quelli che dipendono da script, automazione o CloudFormation modelli, specifica esplicitamente la versione `--engine-version` principale prevista.

Dimensionamento delle istanze

Uno degli aspetti più critici nella scelta della dimensione di un'istanza in Amazon DocumentDB è la quantità di RAM per la cache. Amazon DocumentDB riserva un terzo della RAM per i propri servizi, il che significa che solo due terzi della RAM dell'istanza sono disponibili per la cache. Pertanto, è una best practice di Amazon DocumentDB scegliere un tipo di istanza con RAM sufficiente per contenere in memoria il tuo set di lavoro (ad esempio dati e indici). Avere istanze di dimensioni adeguate aiuterà a ottimizzare le prestazioni complessive e potenzialmente a minimizzare i costi. I/O Puoi utilizzare il calcolatore di dimensionamento di Amazon DocumentDB di terze parti per stimare la dimensione dell'istanza per un particolare carico di lavoro.

Per determinare se il set di lavoro dell'applicazione è compatibile con la memoria, monitora l'`BufferCacheHitRatio`utilizzo di Amazon CloudWatch per ogni istanza in un cluster sottoposto a carico.

La `BufferCacheHitRatio` CloudWatch metrica misura la percentuale di dati e indici forniti dalla cache di memoria di un'istanza (rispetto al volume di archiviazione). In generale, il valore di `BufferCacheHitRatio` dovrebbe essere il più alto possibile, poiché la lettura dei dati dalla memoria del working set è più veloce e più conveniente rispetto alla lettura dal volume di archiviazione. Anche se è consigliabile mantenere il valore `BufferCacheHitRatio` il più vicino possibile al 100%, il miglior valore ottenibile dipenderà dai modelli di accesso e dai requisiti di prestazione dell'applicazione. Per mantenere il massimo valore `BufferCacheHitRatio` possibile , si consiglia di eseguire il provisioning delle istanze del cluster con una sufficiente quantità di RAM per poter adattare gli indici e i working set di dati in memoria.

Se i tuoi indici non si adattano alla memoria, vedrai un valore inferiore `BufferCacheHitRatio`. La lettura continua da disco comporta I/O costi aggiuntivi e non è migliore della lettura dalla memoria. Se il rapporto `BufferCacheHitRatio` è inferiore al previsto, aumentare la dimensione dell'istanza del cluster per fornire più RAM per adattare i dati del working set in memoria. Se il ridimensionamento della classe di istanza si traduce in un aumento drastico di `BufferCacheHitRatio`, il set di lavoro dell'applicazione non si adattava alla memoria. Continua con l'incremento fino a quando il valore di `BufferCacheHitRatio` non aumenta più drasticamente dopo un'operazione di dimensionamento. Per ulteriori informazioni sul monitoraggio dei parametri di un'istanza, consulta Metriche di Amazon DocumentDB.

A seconda del carico di lavoro e dei requisiti di latenza, potrebbe essere accettabile che l'applicazione abbia valori `BufferCacheHitRatio` più elevati durante l'utilizzo della fase costante, ma abbia periodicamente delle riduzioni dei valori `BufferCacheHitRatio` quando le query analitiche che devono eseguire la scansione di un'intera raccolta vengono eseguite su un'istanza. Questi riduzioni periodiche del valore `BufferCacheHitRatio` possono manifestarsi come latenza più elevata per le query successive che devono ripopolare i dati del working set dal volume di archiviazione nella cache del buffer. Testa i tuoi carichi di lavoro in un ambiente di pre-produzione con un carico di lavoro di produzione rappresentativo innanzitutto per comprendere le caratteristiche prestazionali e **`BufferCacheHitRatio`** prima di implementare il carico di lavoro in produzione.

`BufferCacheHitRatio` è un parametro specifici dell'istanza, pertanto istanze diverse all'interno dello stesso cluster possono avere valori `BufferCacheHitRatio` diversi a seconda della modalità di distribuzione delle letture tra le istanze primarie e di replica. Se il carico di lavoro operativo non è in grado di gestire gli aumenti periodici della latenza dal ripopolamento della cache del working set dopo l'esecuzione di query analitiche, ti consigliamo di provare a isolare la cache del buffer del carico di lavoro normale da quella delle query analitiche. Puoi ottenere il completo isolamento di `BufferCacheHitRatio` indirizzando le query operative all'istanza primaria e le query analitiche solo alle istanze di replica. Puoi inoltre ottenere l'isolamento parziale indirizzando le query analitiche a un'istanza di replica specifica, con la consapevolezza che una percentuale di query regolari verrà eseguita anche su tale replica e potrebbe potenzialmente essere influenzata.

I valori appropriati di `BufferCacheHitRatio` dipendono dal caso d'uso e dai requisiti dell'applicazione. Non esiste un valore migliore o un valore minimo per questo parametro; solo tu puoi decidere se il compromesso rappresentato da un valore temporaneamente inferiore di `BufferCacheHitRatio` è accettabile dal punto di vista dei costi e delle prestazioni.

Utilizzo degli indici

Indici degli edifici

Quando importi dati in Amazon DocumentDB, devi creare i tuoi indici prima di importare set di dati di grandi dimensioni. Puoi utilizzare Amazon DocumentDB Index Tool per estrarre gli indici da un'istanza o `mongodump` directory MongoDB in esecuzione e crearli in un cluster Amazon DocumentDB. Per ulteriori informazioni sulle migrazioni, consulta Migrazione e aggiornamento di Amazon DocumentDB.

Selettività degli indici

Limita la creazione dell'indice ai campi in cui il numero di valori duplicati è inferiore all'1% del numero totale di documenti della raccolta. Ad esempio, se la raccolta contiene 100.000 documenti, creare solo indici nei campi in cui lo stesso valore si verifica al massimo 1000 volte.

La scelta di un indice con un numero elevato di valori univoci (ad esempio, un'elevata cardinalità) garantisce che le operazioni di filtro restituiscano un numero ridotto di documenti, ottenendo così buone prestazioni durante le scansioni degli indici. Un esempio di indice ad elevata cardinalità è un indice univoco, che garantisce che i predicati di uguaglianza restituiscano al massimo un singolo documento. Esempi di bassa cardinalità includono un indice su un campo booleano e un indice nel giorno della settimana. A causa delle scarse prestazioni, è improbabile che gli indici a bassa cardinalità vengano scelti dall'ottimizzatore delle query del database. Allo stesso tempo, gli indici a bassa cardinalità continuano a consumare risorse come spazio su disco e. I/Os Come regola generale, occorre indicare indici su campi in cui la frequenza tipica del valore è l'1% della dimensione totale della raccolta.

Inoltre, è consigliabile creare solo indici su campi comunemente utilizzati come un filtro e cercare regolarmente indici inutilizzati. Per ulteriori informazioni, consulta Come posso analizzare l'utilizzo degli indici e identificare gli indici inutilizzati?.

Impatto degli indici sulla scrittura dei dati

Sebbene gli indici possano migliorare le prestazioni delle query evitando la necessità di eseguire la scansione di tutti i documenti di una raccolta, questo miglioramento comporta un compromesso. Per ogni indice di una raccolta, ogni volta che un documento viene inserito, aggiornato o eliminato, il database deve aggiornare la raccolta e scrivere i campi in ciascuno degli indici per la raccolta. Ad esempio, se una raccolta dispone di nove indici, il database deve eseguire dieci scritture prima di

riconoscere l'operazione al client. Pertanto, ogni indice aggiuntivo comporta una latenza di scrittura aggiuntiva e un aumento I/O dello spazio di archiviazione complessivo utilizzato.

Le istanze del cluster devono essere dimensionate in modo appropriato per mantenere tutta la memoria del set di lavoro. Ciò evita la necessità di leggere continuamente le pagine di indice dal volume di archiviazione, il che influisce negativamente sulle prestazioni e genera costi più elevati. I/O Per ulteriori informazioni, consulta Dimensionamento delle istanze.

Per ottenere prestazioni ottimali, ridurre al minimo il numero di indici nelle raccolte, aggiungendo solo gli indici necessari per migliorare le prestazioni per le query comuni. Mentre i carichi di lavoro variano, una buona linea guida consiste nel mantenere il numero di indici per raccolta a cinque o meno.

Identificazione degli indici mancanti

È consigliabile identificare regolarmente gli indici mancanti. Per ulteriori informazioni, consulta Come faccio a identificare gli indici mancanti?.

Identificazione degli indici non utilizzati

Come best practice, è consigliabile identificare e rimuovere regolarmente gli indici inutilizzati. Per ulteriori informazioni, consulta Come posso analizzare l'utilizzo degli indici e identificare gli indici inutilizzati?.

Best practice di sicurezza

Per le migliori pratiche di sicurezza per Amazon DocumentDB, consulta. Le migliori pratiche di sicurezza per Amazon DocumentDB

Ottimizzazione dei costi

Le seguenti best practice possono aiutarti a gestire e ridurre al minimo i costi di utilizzo di Amazon DocumentDB. Per informazioni sui prezzi, consulta le domande frequenti su Amazon DocumentDB (con compatibilità MongoDB) e Amazon DocumentDB (con compatibilità MongoDB).

- Crea avvisi di fatturazione in corrispondenza delle soglie al 50% e 75% della fattura mensile attesa. Per ulteriori informazioni sulla creazione di avvisi di fatturazione, consulta Creazione di un allarme di fatturazione.
- L'architettura di Amazon DocumentDB separa lo storage dall'elaborazione, quindi anche un cluster a istanza singola è estremamente resistente. Il volume di storage del cluster replica i dati in sei modi su tre zone di disponibilità, garantendo una durabilità estremamente elevata

indipendentemente dal numero di istanze nel cluster. Un tipico cluster di produzione dispone di tre o più istanze per fornire la disponibilità elevata. Tuttavia, puoi ottimizzare i costi utilizzando un cluster di sviluppo a istanza singola quando non è richiesta la disponibilità elevata.

- Per scenari di sviluppo e di test, arresta un cluster quando non è più necessario e avvialo quando lo sviluppo riprende. Per ulteriori informazioni, consulta Arresto e avvio di un cluster Amazon DocumentDB.
- Sia il TTL che i flussi di modifica si verificano quando i dati vengono scritti, letti I/O ed eliminati. Se queste funzionalità sono state abilitate ma non vengono utilizzate nell'applicazione, la disattivazione delle funzionalità può contribuire a ridurre i costi.

Utilizzo di parametri per identificare problemi a livello di prestazioni

Argomenti

- Visualizzazione dei parametri relativi alle prestazioni
- Impostazione di un allarme CloudWatch
- Valutazione dei parametri relativi alle prestazioni
- Valutazione dell'utilizzo delle istanze Amazon DocumentDB con metriche CloudWatch
- Ottimizzazione di query

Per identificare i problemi di prestazioni causati da risorse insufficienti e altri colli di bottiglia comuni, puoi monitorare le metriche disponibili per il tuo cluster Amazon DocumentDB.

Visualizzazione dei parametri relativi alle prestazioni

Monitora regolarmente i parametri relativi alle prestazione per osservare i valori medi, massimi e minimi per vari intervalli di tempo. Ciò ti consente di identificare quando le prestazioni subiscono un calo. Puoi anche impostare CloudWatch allarmi Amazon per soglie metriche particolari in modo da essere avvisato se vengono raggiunte.

Per risolvere i problemi relativi alle prestazioni, è importante comprendere le prestazioni di base del sistema. Quando configuri un nuovo cluster e lo esegui con un carico di lavoro tipico, acquisisci il valore medio, massimo e minimo di tutti i parametri relativi alle prestazioni a intervalli diversi (ad esempio, un'ora, 24 ore, una settimana, due settimane). Ciò ti permette di avere un quadro dei valori normali. Ciò aiuta anche a effettuare confronti delle attività durante le ore di punta e non di punta. Puoi quindi utilizzare queste informazioni per identificare quando le prestazioni scendono al di sotto dei livelli standard.

Puoi visualizzare le metriche delle prestazioni utilizzando o. Console di gestione AWS AWS CLI Per ulteriori informazioni, consulta Visualizzazione CloudWatch dei dati.

Impostazione di un allarme CloudWatch

Per impostare un CloudWatch allarme, consulta Using Amazon CloudWatch Alarms nella Amazon CloudWatch User Guide.

Valutazione dei parametri relativi alle prestazioni

Un'istanza ha diverse categorie di parametri. La modalità di determinazione dei valori accettabili dipende dal parametro.

CPU

- Utilizzo della CPU: percentuale della capacità di elaborazione del computer utilizzata.

Memoria

- Memoria liberabile: quanta RAM è disponibile nell'istanza.
- Utilizzo dello swap: quanto spazio di swap viene utilizzato dall'istanza, in megabyte.

Input/output operazioni

- Read IOPS, Write IOPS: il numero medio di operazioni di lettura o scrittura su disco al secondo.
- Latenza di lettura, latenza di scrittura: il tempo medio di un'operazione di lettura o scrittura in millisecondi.
- Velocità effettiva di lettura, velocità effettiva di scrittura: il numero medio di megabyte letti o scritti su disco al secondo.
- Profondità della coda del disco: il numero di I/O operazioni in attesa di essere scritte o lette dal disco.

Traffico di rete

- Throughput di ricezione di rete, Throughput di trasmissione di rete: la velocità del traffico di rete da e verso l'istanza in megabyte al secondo.

Connessioni database

- Connessioni DB: il numero di sessioni client connesse all'istanza.

In generale, i valori accettabili per i parametri relativi alle prestazioni dipendono dalla baseline e dall'attività dell'applicazione. Indagare le variazioni della baseline coerenti o che rappresentano dei trend.

Di seguito sono riportati alcuni suggerimenti su tipi di parametri specifici:

- Consumo elevato di CPU: valori elevati per il consumo di CPU potrebbero essere appropriati, a condizione che siano in linea con gli obiettivi dell'applicazione (come la velocità effettiva o la concorrenza) e siano previsti. Se il consumo della CPU supera costantemente l'80%, valuta la possibilità di aumentare le dimensioni delle istanze.
- Consumo elevato di RAM: se la `FreeableMemory` metrica scende spesso al di sotto del 10% della memoria totale dell'istanza, valuta la possibilità di aumentare le istanze. Per ulteriori informazioni su cosa succede quando l'istanza DocumentDB presenta una pressione di memoria elevata, consulta Amazon DocumentDB Resource Governance.
- Utilizzo degli swap: questa metrica deve rimanere pari o prossima allo zero. Se l'utilizzo di swap è significativo, valuta la possibilità di aumentare le dimensioni delle istanze.
- Traffico di rete: per il traffico di rete, contatta l'amministratore di sistema per capire qual è il throughput previsto per la rete di dominio e la connessione Internet. Indaga il traffico di rete se il throughput è costantemente al di sotto del valore previsto.
- Connessioni al database: valuta la possibilità di limitare le connessioni al database se riscontri un numero elevato di connessioni utente e una diminuzione delle prestazioni delle istanze e dei tempi di risposta. Il numero ideale di connessioni utente per l'istanza dipende dalla classe di istanza e dalla complessità delle operazioni eseguite. In caso di problemi con i parametri relativi alle prestazioni, per migliorare la situazione puoi provare a ottimizzare le query più utilizzate e costose per verificare se ciò riduce la pressione sulle risorse di sistema.

Se le tue query sono ottimizzate e il problema persiste, valuta la possibilità di aggiornare la tua classe di istanza Amazon DocumentDB a una classe con più risorse (CPU, RAM, spazio su disco, larghezza di banda di rete, I/O capacità) correlate al problema riscontrato.

Valutazione dell'utilizzo delle istanze Amazon DocumentDB con metriche CloudWatch

Puoi utilizzare le CloudWatch metriche per monitorare il throughput dell'istanza e scoprire se la classe di istanze fornisce risorse sufficienti per le tue applicazioni. Per informazioni sulle quote delle classi di istanze, consulta Quote delle istanze e individua le specifiche della classe di istanza per determinare le prestazioni della tua rete.

Se l'utilizzo dell'istanza è vicino al limite della classe di istanza, le prestazioni potrebbero iniziare a rallentare. Le CloudWatch metriche possono confermare questa situazione, quindi puoi pianificare di passare manualmente a una classe di istanze più ampia.

Combina i seguenti valori CloudWatch delle metriche per scoprire se ti stai avvicinando al limite della classe di istanze:

- NetworkThroughput—La quantità di throughput di rete ricevuta e trasmessa dai client per ogni istanza nel cluster Amazon DocumentDB. Questo valore di throughput non include il traffico di rete tra le istanze del cluster e il volume di storage del cluster.
- StorageNetworkThroughput—La quantità di throughput di rete ricevuta e inviata al volume di storage del cluster Amazon DocumentDB da ciascuna istanza del cluster Amazon DocumentDB.

Aggiungilo NetworkThroughput a per trovare il StorageNetworkThroughput throughput di rete ricevuto e inviato al volume di storage del cluster Amazon DocumentDB da ciascuna istanza del tuo cluster Amazon DocumentDB. Il limite della classe dell'istanza deve essere maggiore della somma di questi due parametri combinati.

È possibile utilizzare i seguenti parametri per esaminare ulteriori dettagli del traffico di rete proveniente dalle applicazioni client durante l'invio e la ricezione:

- NetworkReceiveThroughput—La quantità di throughput di rete ricevuta dai client da ciascuna istanza nel cluster Amazon DocumentDB. Questo throughput non include il traffico di rete tra le istanze del cluster e il volume di storage del cluster.
- NetworkTransmitThroughput—La quantità di throughput di rete inviata ai client da ciascuna istanza del cluster Amazon DocumentDB. Questo throughput non include il traffico di rete tra le istanze del cluster e il volume di storage del cluster.
- StorageNetworkReceiveThroughput—La quantità di throughput di rete ricevuta dal volume di storage del cluster Amazon DocumentDB da ciascuna istanza del cluster.
- StorageNetworkTransmitThroughput—La quantità di throughput di rete inviata al volume di storage del cluster Amazon DocumentDB da ciascuna istanza del cluster.

Somma tutti questi parametri per confrontare l'utilizzo della rete e il limite della classe di istanza. Il limite della classe di istanza deve essere maggiore della somma di questi parametri combinati.

I limiti di rete e l'utilizzo della CPU da parte di un'istanza sono reciproci. quando la velocità di trasmissione effettiva della rete aumenta, cresce anche l'utilizzo della CPU. Il monitoraggio dell'utilizzo della CPU e della rete fornisce informazioni su come e perché le risorse vengono esaurite.

Per ridurre al minimo l'utilizzo della rete, puoi prendere in considerazione:

- L'uso di una classe di istanza più grande.
- La suddivisione delle richieste di scrittura in batch per ridurre le transazioni in generale.
- L'instradamento del carico di lavoro di sola lettura a un'istanza di sola lettura.
- L'eliminazione degli indici non utilizzati.

Ottimizzazione di query

Uno dei modi migliori per migliorare le prestazioni di un cluster consiste nell'ottimizzare le query più comuni e a uso più intensivo di risorse per renderle meno costose da eseguire.

Puoi utilizzare il profiler (vedi Profilazione delle operazioni di Amazon DocumentDB) per registrare il tempo di esecuzione e i dettagli delle operazioni eseguite nel cluster. Il profiler è utile per monitorare le operazioni più lente sul cluster per aiutare a migliorare le prestazioni delle singole query e le prestazioni complessive del cluster.

Puoi inoltre utilizzare il comando `explain` per informazioni su come analizzare un piano di query per una determinata query. Utilizza queste informazioni per modificare una query o una raccolta sottostante in modo da migliorare le prestazioni della query (ad esempio, aggiungendo un indice).

Carichi di lavoro TTL e serie temporali

L'eliminazione del documento risultante dalla scadenza dell'indice TTL è un processo di best effort. L'eliminazione dei documenti entro un termine specifico non è garantita. Fattori come la dimensione dell'istanza, l'utilizzo delle risorse dell'istanza, la dimensione del documento, il throughput complessivo, il numero di indici e il fatto che gli indici e il working set si adattino o meno nella memoria possono influenzare i tempi di eliminazione dei documenti scaduti dal processo TTL.

Quando il monitor TTL elimina i documenti, ogni cancellazione comporta dei I/O costi, il che comporta un aumento della fattura. Se la velocità effettiva e i tassi di eliminazione TTL aumentano, dovresti

aspettarti una fattura più elevata a causa del maggiore utilizzo. I/O Tuttavia, se non si crea un indice TTL per eliminare i documenti, ma si segmentano i documenti in raccolte in base al tempo e si eliminano semplicemente le raccolte quando non sono più necessarie, non si incorre in alcun costo di I/O. Questo può essere molto più conveniente rispetto all'utilizzo di un indice TTL.

Per i carichi di lavoro di serie temporali, puoi prendere in considerazione la creazione di raccolte continue anziché di un indice TTL, poiché le raccolte cumulative possono essere un modo migliore e meno impegnativo per eliminare i dati. I/O Se disponi di raccolte di grandi dimensioni (in particolare raccolte superiori a 1 TB) o I/O i costi di eliminazione TTL sono un problema, suddividi i documenti in raccolte in base al tempo ed elimina le raccolte quando i documenti non sono più necessari. Puoi creare una raccolta al giorno o alla settimana, a seconda della frequenza di inserimento dei dati. Anche se i requisiti variano a seconda dell'applicazione, una buona regola generale è quella di avere raccolte più piccole piuttosto che alcune raccolte di grandi dimensioni. L'eliminazione di queste raccolte non comporta I/O costi e può essere più rapida e conveniente rispetto all'utilizzo di un indice TTL.

Migrazioni

Come best practice per la migrazione dei dati su Amazon DocumentDB, crea prima i tuoi indici in Amazon DocumentDB e poi migra i tuoi dati. Creando per primi gli indici, si riduce il tempo complessivo e si aumenta la velocità della migrazione. A tale scopo, puoi utilizzare Amazon DocumentDB Index Tool. https://github.com/awslabs/amazon-documentdb-tools Per ulteriori informazioni sulle migrazioni, consulta il. Migrazione e aggiornamento di Amazon DocumentDB

Si consiglia inoltre, prima di migrare il database di produzione, di testare completamente l'applicazione su Amazon DocumentDB, prendendo in considerazione funzionalità, prestazioni, operazioni e costi.

Lavorare con i gruppi di parametri del cluster

Prova le modifiche ai gruppi di parametri del cluster su un cluster di test prima di applicarle ai cluster di produzione. Per ulteriori informazioni sul backup di un cluster, consulta Backup e ripristino in Amazon DocumentDB.

Interrogazioni sulla pipeline di aggregazione

Quando crei una query di pipeline di aggregazione con più fasi e valuti solo un sottoinsieme dei dati nella query, utilizza la fase `$match` come prima fase o all'inizio della pipeline. Utilizzando prima

`$match` ridurrai il numero di fasi successive dei documenti all'interno della query di pipeline di aggregazione che sarà necessario elaborare, migliorando così le prestazioni della query.

`BatchInsert e BatchUpdate`

Quando si esegue una frequenza elevata di `batchInsert` and/or `batchUpdate` operazioni simultanee e la quantità di `FreeableMemory` (CloudWatch Metric) va a zero sull'istanza principale, è possibile ridurre la concorrenza del carico di lavoro di inserimento o aggiornamento del batch oppure, se la concorrenza del carico di lavoro non può essere ridotta, aumentare la dimensione dell'istanza per aumentare la quantità di. `FreeableMemory`

Gestione degli indici Amazon DocumentDB

Argomenti

- Creazione di indici Amazon DocumentDB
- Gestione degli indici Amazon DocumentDB

Creazione di indici Amazon DocumentDB

La creazione di indici in Amazon DocumentDB richiede una serie di decisioni:

- Quanto velocemente deve essere completato?
- La raccolta può essere inaccessibile durante la compilazione?
- Quanta potenza di calcolo di un'istanza può essere allocata alla build?
- Che tipo di indice deve essere creato?

Questa sezione ti aiuta a rispondere a queste domande e fornisce i comandi e gli esempi di monitoraggio per creare un indice Amazon DocumentDB sulla tua raccolta di cluster basata su istanze.

Linee guida

Le seguenti linee guida includono limiti di base e compromessi di configurazione nella creazione di nuovi indici:

- Supporto delle versioni di Amazon DocumentDB: mentre l'indicizzazione di un singolo lavoratore è supportata in tutte le versioni di Amazon DocumentDB, l'indicizzazione di più lavoratori è supportata solo nelle versioni 4.0, 5.0 e 8.0 di Amazon DocumentDB.
- Compromesso in termini di prestazioni: l'aumento del numero di lavoratori nel processo di creazione dell'indice aumenta l'utilizzo della CPU e l'IO di lettura sull'istanza principale del database Amazon DocumentDB. Le risorse necessarie per creare un nuovo indice non saranno disponibili per il carico di lavoro in esecuzione.
- Cluster elastici: l'indicizzazione parallela non è supportata sui cluster elastici Amazon DocumentDB.
- Cluster serverless: l'indicizzazione parallela su cluster serverless richiede una capacità minima di almeno 32 DCU. Al di sotto di questa soglia, il `workers` parametro è accettato ma la creazione dell'indice viene eseguita con un singolo worker. Il numero massimo di lavoratori paralleli varia in base alla capacità minima configurata: sono disponibili circa `min_DCU / 8` lavoratori. Ad esempio, con 32 DCU è possibile utilizzare fino a 4 lavoratori e con 64 DCU è possibile utilizzare fino a 8 lavoratori.

  > **Note**
  >
  > È necessario riavviare il cluster dopo aver modificato l'impostazione della capacità minima per rendere effettivo il nuovo limite di lavoratori paralleli.

- Numero massimo di lavoratori (basato sull'istanza): il numero massimo di lavoratori che è possibile configurare dipende dalle dimensioni dell'istanza principale nel cluster di database. È la metà del numero totale di vCPU sull'istanza primaria del cluster di database. Ad esempio, è possibile eseguire un massimo di 32 worker su un'istanza db.r6g.16xlarge con 64 vCPU.

  > **Note**
  >
  > I lavoratori paralleli sono supportati solo sulle istanze 2xlarge e successive.

- Numero minimo di lavoratori: il numero minimo di lavoratori che puoi configurare è uno. L'impostazione predefinita per la creazione dell'indice su cluster basati su istanze è di due worker. Tuttavia, è possibile ridurre il numero di lavoratori a uno utilizzando l'opzione «worker threads». Questo eseguirà il processo con un solo lavoratore.
- Compressione dell'indice: Amazon DocumentDB non supporta la compressione dell'indice. Le dimensioni dei dati per gli indici potrebbero essere maggiori rispetto a quando si utilizzano altre opzioni.

- Indicizzazione di più raccolte: metà delle vCPU sull'istanza principale del cluster di database può essere utilizzata per i lavoratori configurati che eseguono la creazione di indici su più raccolte.
- Tipi di indice: per una spiegazione completa dei tipi di indice supportati su Amazon DocumentDB, consulta Come indicizzare su Amazon DocumentDB nel blog del database. AWS

Nozioni di base

Per iniziare la creazione dell'indice su una raccolta, usa il `createIndexes` comando. Per impostazione predefinita, il comando eseguirà due worker paralleli che aumentano la velocità del processo di creazione dell'indice di due volte.

Ad esempio, il seguente processo di comando dimostra come creare un indice per il campo «user_name» in un documento e aumentare la velocità del processo di indicizzazione a quattro worker:

1. Crea indici utilizzando due worker paralleli sul cluster:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx"}]})
```

2. Per ottimizzare la velocità del processo di creazione dell'indice, è possibile specificare il numero di lavoratori utilizzando l'opzione «worker threads» (`"workers":<number>`) nel comando. `db.runCommand createIndexes`

   Aumentate la velocità del processo fino a quattro lavoratori paralleli:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "workers":4}]})
```

> **Note**
>
> Maggiore è il numero di lavoratori, più velocemente procede la creazione dell'indice. Tuttavia, maggiore è il numero di lavoratori, maggiore è il carico sulle vCPU e sull'IO di lettura dell'istanza primaria. Assicurati che il cluster sia sufficientemente predisposto per gestire l'aumento del carico senza compromettere gli altri carichi di lavoro.

Stato di avanzamento dell'indicizzazione

Il processo di creazione dell'indice funziona inizializzando, analizzando le raccolte, ordinando le chiavi e, infine, inserendo le chiavi tramite un generatore di indici. Il processo prevede fino a sei fasi quando viene eseguito in primo piano e fino a nove fasi quando viene eseguito in background. È possibile visualizzare le metriche sullo stato, ad esempio la percentuale di completamento, il numero totale di blocchi di archiviazione scansionati, le chiavi ordinate e le chiavi inserite, fase per fase.

Monitora l'avanzamento del processo di indicizzazione utilizzando il `db.currentOp()` comando nella shell mongo. Il completamento al 100% dell'ultima fase mostra che tutti gli indici sono stati creati con successo:

```
db.currentOp({"command.createIndexes": { $exists : true } })
```

> **Note**
>
> La visualizzazione dello stato di avanzamento dell'indicizzazione è supportata solo su Amazon DocumentDB 5.0.

Tipi di compilazione dell'indice

I quattro tipi di build dell'indice sono:

- Foreground: la build dell'indice in primo piano blocca tutte le altre operazioni del database fino alla creazione dell'indice. La build in primo piano di Amazon DocumentDB è composta da cinque fasi.
- Foreground (unique): le build con indice in primo piano a documento singolo (univoco) bloccano altre operazioni del database, ad esempio le normali compilazioni in primo piano. A differenza della build di base in primo piano, la build univoca utilizza una fase aggiuntiva (ordinamento delle chiavi 2) per cercare le chiavi duplicate. La build in primo piano (unica) è composta da sei fasi.
- Background: la creazione dell'indice in background consente di eseguire altre operazioni del database in primo piano durante la creazione dell'indice. La build in background di Amazon DocumentDB è composta da otto fasi.
- Background (univoco): le compilazioni di indici in background per un singolo documento (univoco) consentono l'esecuzione di altre operazioni del database in primo piano durante la creazione dell'indice. A differenza della build di base in background, la build univoca utilizza una fase aggiuntiva (ordinamento delle chiavi 2) per cercare le chiavi duplicate. La build di sfondo (unica) è composta da nove fasi.

Fasi di creazione dell'indice

Fase	Primo piano	Primo piano (unico)	Contesto	Sfondo (unico)
Inizializzazione	1	1	1	1
indice di costruzione: inizializzazione	2	2	2	2
indice di costruzione: raccolta di scansioni	3	3	3	3
indice degli edifici: chiavi di ordinamento 1	4	4	4	4
indice di costruzio ne: chiavi di ordinamento 2		5		5
indice di costruzione: inserimento di chiavi	5	6	5	6
convalida: indice di scansione			6	7
convalida: ordinamento delle tuple			7	8

Fase	Primo piano	Primo piano (unico)	Contesto	Sfondo (unico)
convalida: scansione della raccolta			8	9

- inizializzazione: createIndex sta preparando il generatore di indici. Questa fase dovrebbe essere molto breve.
- building index: inizializzazione - Il generatore di indici si sta preparando a creare l'indice. Questa fase dovrebbe essere molto breve.
- building index: scanning collection - Il generatore di indici sta eseguendo una scansione della raccolta per raccogliere le chiavi dell'indice. L'unità di misura è «blocchi».

> **Note**
>
> Se più di un lavoratore è configurato per la creazione dell'indice, questo viene visualizzato in questa fase. La fase di «scansione della raccolta» è l'unica fase che utilizza più worker durante il processo di creazione dell'indice. Tutte le altre fasi mostreranno un singolo lavoratore.

- building index: ordinamento delle chiavi 1 - Il generatore di indici sta ordinando le chiavi di indice raccolte. L'unità di misura è «chiavi».
- building index: sorting keys 2 - Il generatore di indici ordina le chiavi di indice raccolte che corrispondono a tuple morte. Questa fase esiste solo per la creazione di un indice univoco. L'unità di misura è «chiavi».
- building index: inserimento di chiavi - Il generatore di indici sta inserendo le chiavi dell'indice nel nuovo indice. L'unità di misura è «chiavi».
- convalida: indice di scansione - CreateIndex sta scansionando l'indice per trovare le chiavi che devono essere convalidate. L'unità di misura è «blocchi».
- validazione: ordinamento delle tuple - CreateIndex sta ordinando l'output della fase di scansione dell'indice.
- convalida: scansione della raccolta - CreateIndex sta scansionando la raccolta per convalidare le chiavi di indice trovate nelle due fasi precedenti. L'unità di misura è «blocchi».

Esempio di output per la creazione dell'indice

Nel seguente esempio di output (creazione dell'indice in primo piano), viene mostrato lo stato della creazione dell'indice. Il campo «msg» riassume l'avanzamento della compilazione indicando la fase e la percentuale di completamento della compilazione. Il campo «workers» indica il numero di lavoratori utilizzati durante quella fase della creazione dell'indice. Il campo «avanzamento» mostra i numeri effettivi utilizzati per calcolare la percentuale di completamento.

> **Note**
>
> I campi «currentIndexBuildName», «msg» e «progress» non sono supportati in Amazon DocumentDB versione 4.0.

```
{
    "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
}
```

Gestione degli indici Amazon DocumentDB

Argomenti

- Index bloat
- Manutenzione dell'indice utilizzando ReIndex

Index bloat

Amazon DocumentDB utilizza Multi-Version Concurrency Control (MVCC) per gestire transazioni simultanee. Quando i documenti vengono eliminati o aggiornati, le versioni precedenti rimangono nelle raccolte e negli indici come versioni «morte». Il processo di garbage collection recupera automaticamente lo spazio da queste versioni obsolete per operazioni future.

L'ingrandimento dell'indice si verifica quando gli indici di una raccolta diventano più grandi a causa dell'accumulo di voci di indice morte o obsolete o della frammentazione all'interno delle pagine. La percentuale riportata rappresenta la quantità di spazio nell'indice che può essere utilizzata dalle voci future dell'indice. Questo ingrossamento consuma spazio sia nella cache del buffer che nella memoria. Se desideri rimuovere il gonfiore, dovrai ricostruire gli indici.

Example Esempio

Esegui il comando seguente per determinare lo spazio di archiviazione inutilizzato per il tuo indice:

```
db.coll.aggregate({$indexStats:{}});
```

Il risultato è simile a questo:

```
{
    "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
    }
}
```

È possibile ricostruire gli indici senza tempi di inattività utilizzando il `reIndex` comando, che richiede una scansione dell'intera raccolta. Consulta Manutenzione dell'indice utilizzando ReIndex .

Manutenzione dell'indice utilizzando ReIndex

`reIndex`è un comando usato per ricostruire un indice. Viene in genere utilizzato quando un indice è danneggiato o inefficiente. Nel tempo, gli indici possono accumulare spazio inutilizzato a causa di numerosi aggiornamenti, inserimenti o eliminazioni, con conseguente riduzione delle prestazioni. La reindicizzazione aiuta a rimuovere lo spazio inutilizzato e a ripristinare l'efficienza dell'indice.

Linee guida ReIndex

- `reIndex`è supportato solo su Amazon DocumentDB 5.0.
- Amazon DocumentDB supporta `reindex` un singolo indice in background, che consente l'utilizzo di più lavoratori. Il vecchio indice è utilizzabile tramite query durante l'esecuzione del `reIndex` processo.
- Amazon DocumentDB supporta l'indicizzazione del rapporto sullo stato di avanzamento tramite. `currentOp` Puoi vedere fasi di creazione dell'indice simili a quelle Fasi di creazione dell'indice visualizzate durante la creazione dell'indice. L'unica differenza è che ha `reIndex` sempre otto fasi, indipendentemente dal fatto che sia unico o meno. Non esiste una fase «indice di costruzione: ordinamento delle chiavi 2».
- `reIndex`può essere eseguito contemporaneamente a qualsiasi comando ad eccezione dei comandi relativi all'indice sulla stessa raccolta:`createIndexes`,, `dropIndexes` e. `collMod` `renameCollection`
- `reIndex`attualmente non è supportato per gli indici testuali, geospaziali, vettoriali e parziali.

`compilazione ReIndex`

Usa il seguente comando per ricostruire il tuo indice:

```
db.runCommand({ reIndex: "collection-name", index: "index-name"})
```

Facoltativamente, puoi anche controllare il numero di lavoratori assegnati al processo di ricostruzione:

```
db.runCommand({ reIndex: "collection-name", index: "index-name", workers: number })
```

Gestione della compressione dei documenti a livello di raccolta

La compressione dei documenti a livello di raccolta di Amazon DocumentDB 5.0 consente di ridurre i costi di storage e I/O comprimendo i documenti nelle raccolte. Puoi abilitare la compressione dei documenti a livello di raccolta e visualizzare i parametri di compressione in base alle esigenze misurando i vantaggi in termini di archiviazione grazie a parametri di compressione come le dimensioni di archiviazione dei documenti compressi e lo stato di compressione. Amazon DocumentDB 5.0 utilizza l'algoritmo di compressione LZ4 per comprimere i documenti.

Amazon DocumentDB supporta la compressione dei documenti a partire dalla versione 5.0. Le seguenti sono funzioni di compressione dei documenti a livello di raccolta:

- Comportamento predefinito: l'impostazione di compressione predefinita per le nuove raccolte in un cluster 5.0 è determinata dal parametro cluster. `default_collection_compression` Per impostazione predefinita, questo parametro è impostato su «disabilitato».
- Compressione delle raccolte esistenti: l'impostazione di compressione per le raccolte esistenti può essere modificata utilizzando il `collMod` comando.
- Modifica della soglia di compressione: la soglia di compressione predefinita è di 2 KB. Questo valore può essere specificato per le nuove raccolte utilizzando il `createCollection` comando e modificato per le raccolte esistenti utilizzando `collMod` il comando.

> **Note**
>
> La compressione dei documenti di Amazon DocumentDB non è supportata nelle versioni 3.6 e 4.0 di Amazon DocumentDB.

Argomenti

- Gestione della compressione dei documenti
- Monitoraggio della compressione dei documenti

Gestione della compressione dei documenti

Attivazione della compressione dei documenti in una raccolta

Abilita la compressione dei documenti durante la creazione di una raccolta su Amazon DocumentDB 5.0 utilizzando `db.createCollection()` il metodo:

```
db.createCollection( sample_collection,{
    storageEngine : {
        documentDB: {
            compression:{enable: <true | false>}
        }
    }
})
```

Attivazione della compressione dei documenti in un cluster

La compressione dei documenti può essere abilitata per impostazione predefinita per tutte le nuove raccolte a livello di cluster impostando il parametro `default_collection_compression` cluster su «enabled». Quando questo parametro è impostato su «abilitato», le nuove raccolte create nel cluster avranno la compressione abilitata per impostazione predefinita con una soglia di compressione di 2 KB.

Compressione delle raccolte esistenti

È inoltre possibile modificare le impostazioni di compressione per una raccolta esistente utilizzando l'`collMod`operazione e specificando la seguente `storageEngine` configurazione. Nota che la modifica apportata utilizzando questo comando si applicherà solo ai nuovi inserted/updated documenti e la compressione sui documenti inseriti in precedenza non cambierà.

```
db.runCommand({
    collMod: "orders",
    storageEngine: {
        documentDB: {compression: {enable: <true | false>} }
    }
```

```
})
```

Impostazione delle soglie di compressione

Per impostazione predefinita, la soglia di compressione per le raccolte compresse è di 2032 byte. Questo valore di soglia può essere impostato nel `createCollection` comando quando si crea una nuova raccolta con la compressione abilitata:

```
db.createCollection( sample_collection, {
    storageEngine : {
        documentDB: {
            compression: {
                enable: true,
                threshold: <128 - 8000>
            }
        }
    }
})
```

È inoltre possibile modificare la soglia di compressione per una raccolta compressa esistente utilizzando l'`collMod`operazione e specificando la seguente configurazione: `storageEngine`

```
db.runCommand({
    collMod: "orders",
    storageEngine: {
        documentDB: {
            compression: {
                enable: true,
                threshold: <128 - 8000>
            }
        }
    }
})
```

Notate che la soglia di compressione può essere impostata solo su un valore compreso tra 128 e 8000 byte. Inoltre, l'`enable`opzione deve essere impostata su «true» quando si specifica la soglia di compressione.

Monitoraggio della compressione dei documenti

Puoi verificare se una raccolta è compressa e calcolarne il rapporto di compressione come segue.

Visualizza le statistiche di compressione eseguendo il `db.collection.stats()` comando `db.printCollectionStats()` or dalla shell mongo. L'output mostra le dimensioni originali e le dimensioni compresse che è possibile confrontare per analizzare i guadagni di spazio di archiviazione derivanti dalla compressione dei documenti. In questo esempio, le statistiche per una raccolta denominata «sample_collection» sono mostrate di seguito. Di seguito viene utilizzato un fattore di scala di 1024*1024 per generare i valori and in MB. `size` `storageSize`

```
db.sample_collection.stats(1024*1024)
```

Di seguito è riportato un esempio dell'output del comando precedente:

```
{
    "ns" : "test.sample_collection",
    "count" : 1000000,
    "size" : 3906.3,
    "avgObjSize" : 4096,
    "storageSize" : 1953.1,
    compression:{"enabled" : true,"threshold" : 2032},
    ...
}
```

- size - La dimensione originale della raccolta di documenti.
- avg ObjSize - La dimensione media del documento prima della compressione è arrotondata al primo decimale. L'unità di misura è il byte.
- StorageSize: la dimensione di archiviazione della raccolta dopo la compressione. L'unità di misura è il byte.
- abilitato: indica se la compressione è abilitata o disabilitata.

Per calcolare il rapporto di compressione effettivo, dividi la dimensione della raccolta per la dimensione di archiviazione (size/storageSize). Per l'esempio precedente, il calcolo è 3906. 3/1953.1 che si traduce in un rapporto di compressione 2:1.

Gestione della compressione basata su dizionari in Amazon DocumentDB 8.0

Amazon DocumentDB 8.0 introduce un nuovo algoritmo di compressione dei documenti, zstd, come alternativa migliorata a LZ4. Quando abiliti la compressione del dizionario su una raccolta Amazon

DocumentDB 8.0 selezionando Zstd come algoritmo di compressione, i documenti della tua raccolta vengono campionati per creare un dizionario di compressione personalizzato. I documenti nuovi e aggiornati vengono compressi utilizzando questo dizionario e l'algoritmo zstd. Questo approccio consente in genere di ottenere rapporti di compressione più elevati rispetto ai metodi di compressione standard, in particolare per le raccolte con schemi di documenti coerenti o nomi di campo ripetuti.

Lz4 è un algoritmo progettato per una compressione e una decompressione rapide. È noto per essere leggero sulla CPU pur ottenendo una compressione evidente. Zstd è un algoritmo generico che con le impostazioni predefinite utilizza più CPU ma raggiunge rapporti di compressione migliori rispetto a lz4. L'uso dei dizionari migliora ulteriormente la compressione per la maggior parte dei documenti JSON. Alcuni vantaggi dell'algoritmo Zstd sono:

- Costi di archiviazione ridotti: rapporti di compressione più elevati significano un minore utilizzo dello storage e costi inferiori.
- Inferiore I/O: i documenti compressi richiedono meno I/O, con un potenziale miglioramento delle prestazioni.
- Ottimizzato per la tua raccolta: il dizionario è addestrato specificamente per i modelli di dati della tua raccolta.

> **Note**
>
> Dictionary-based la compressione non è supportata nelle versioni 3.6, 4.0 e 5.0 di Amazon DocumentDB.

Considerazioni sulle prestazioni

La compressione Zstd comporta i seguenti compromessi:

- Storage vs. CPU: la compressione Zstd raggiunge rapporti di compressione migliori ma può utilizzare un po' più risorse della CPU rispetto alla compressione LZ4.
- Compressione iniziale: le nuove raccolte potrebbero non raggiungere una compressione ottimale finché non vengono inseriti abbastanza documenti per addestrare un dizionario efficace. Attualmente, un dizionario viene addestrato se raccoglie almeno 100 documenti.
- Tipo di carico di Read-intensive lavoro: i carichi di lavoro in cui tutti i dati vengono inseriti nella cache buffer possono registrare un aumento della latenza e dell'utilizzo della CPU a causa del sovraccarico di decompressione.

La compressione Zstd è particolarmente efficace per le raccolte con documenti di piccole dimensioni, matrici di documenti e nomi di campo ripetuti.

Attivazione della compressione basata su dizionari

Per le nuove raccolte puoi usare il comando seguente per abilitare la compressione Zstd:

```
db.createCollection("myCollection",
    {
        storageEngine: {
            documentDB: {
                compression: {
                    enable: true,
                    algorithm: "zstd"
                }
            }
        }
    }
 )
```

Puoi anche abilitare o modificare la compressione su una raccolta esistente:

```
db.runCommand({
    collMod: "myCollection",
    storageEngine: {
        documentDB: {
            compression: {
                enable: true,
                algorithm: "zstd"
            }
        }
    }
 })
```

Per abilitare l'algoritmo Zstd in tutte le raccolte del cluster, puoi modificare il gruppo di parametri del cluster per selezionare «zstd» come valore per il parametro «default_collection_compression».

Nozioni di base

Amazon DocumentDB 8.0 viene fornito con la compressione Zstd attivata per impostazione predefinita. Puoi sempre disattivarla impostando il valore di 'default_compression' su disabilitato nel gruppo di parametri del cluster. Va notato che a partire da Amazon DocumentDB 8.0, 'enabled' non è più una scelta valida per default_compression ed è necessario selezionare tra Zstd e LZ4.

Monitoraggio

È possibile visualizzare le informazioni di compressione per una raccolta utilizzando uno dei seguenti comandi:

- db.runCommand ({collStats: «myCollection»}) OPPURE
- db.collection.stats ()

Questi comandi restituiscono statistiche chiave che è possibile utilizzare per calcolare il rapporto di compressione:

- compression.algorithm: L'algoritmo utilizzato («lz4" o «zstd»)
- StorageSize: lo spazio di archiviazione effettivo utilizzato dalla raccolta, dopo la compressione. Nota che questo numero include la frammentazione (ovvero lo spazio inutilizzato nelle pagine del database)
- avgObjSize: La dimensione logica media dei documenti della raccolta, decompressi. Tieni presente che se la tua raccolta contiene più di 20.000 documenti, questo numero sarà un'approssimazione basata su un campione di 20.000 documenti.
- dimensione: la dimensione logica della raccolta senza compressione. Questo numero si ottiene moltiplicando la media ObjSize per il numero totale di documenti nella raccolta, quindi se avg ObjSize è un'approssimazione, anche questo numero sarà un'approssimazione.
- conteggio: numero di documenti nella raccolta

Le seguenti CloudWatch metriche possono essere utili per valutare la compressione basata sul dizionario:

- CPUUtilization
- FreeableMemory
- VolumeBytesUsed

- VolumeReadIOPs
- VolumeWriteIOPs

Metriche CollStats:

- storageSize
- formato

Inoltre, può essere utile tenere traccia delle metriche specifiche dell'applicazione, come la latenza e il throughput per tipo di query o API.

Sicurezza in Amazon DocumentDB

La sicurezza del cloud AWS è la massima priorità. Come AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS voi. Questa documentazione ti aiuta a capire come applicare il modello di responsabilità condivisa quando usi Amazon DocumentDB. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che esegue AWS i servizi nel AWS cloud. AWS ti fornisce anche servizi che puoi utilizzare in sicurezza. Third-party gli auditor testano e verificano regolarmente l'efficacia della nostra sicurezza nell'ambito dei programmi di AWS conformità. Per informazioni sui programmi di conformità che si applicano ad Amazon DocumentDB (con compatibilità con MongoDB), consulta AWS Services in Scope by Compliance Program.
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. L'utente è anche responsabile per altri fattori, tra cui la riservatezza dei dati, i requisiti dell'azienda e leggi e normative applicabili.

Amazon DocumentDB è autorizzato ai sensi del Federal Risk and Authorization Management Program (FedRAMP). Ha l'autorizzazione FedRAMP High per le regioni AWS GovCloud (USA) e l'autorizzazione FedRAMP Moderate per le regioni degli Stati Uniti. AWS East/West Per dettagli AWS e iniziative in materia di conformità, vedi AWS Services in Scope by Compliance Program.

> **Note**
>
> Questo capitolo si applica sia ai cluster basati su istanze che ai cluster elastici. Per ulteriori informazioni, consulta gli argomenti riportati di seguito.

Scopri anche come utilizzare altri AWS servizi che ti aiutano a monitorare e proteggere le tue risorse Amazon DocumentDB. I seguenti argomenti mostrano come configurare Amazon DocumentDB per soddisfare i tuoi obiettivi di sicurezza e conformità.

Argomenti

- Gestione delle password con Amazon DocumentDB e AWS Secrets Manager

- Protezione dei dati in Amazon DocumentDB
- Gestione delle identità e degli accessi per Amazon DocumentDB
- Cross-service prevenzione sostitutiva confusa
- Autenticazione tramite identità IAM
- Autenticazione tramite SCRAM-SHA-256
- Gestione degli utenti di Amazon DocumentDB
- Accesso al database tramite Access Control Role-Based
- Aggiornamento dei certificati TLS di Amazon DocumentDB
- Convalida della conformità in Amazon DocumentDB
- Resilienza in Amazon DocumentDB
- Sicurezza dell'infrastruttura in Amazon DocumentDB
- API Amazon DocumentDB e interfaccia (endpoint VPC) (AWS PrivateLink)
- Le migliori pratiche di sicurezza per Amazon DocumentDB
- Verifica degli eventi di Amazon DocumentDB
- Amazon VPC e Amazon DocumentDB

Gestione delle password con Amazon DocumentDB e AWS Secrets Manager

Amazon DocumentDB si integra con Secrets Manager per gestire le password degli utenti primari per i tuoi cluster.

Argomenti

- Limitazioni per l'integrazione di Secrets Manager con Amazon DocumentDB
- Panoramica sulla gestione delle password degli utenti primari con AWS Secrets Manager
- Applicazione della gestione Amazon DocumentDB della password utente principale in AWS Secrets Manager
- Gestione della password utente principale per un cluster con Secrets Manager

Limitazioni per l'integrazione di Secrets Manager con Amazon DocumentDB

La gestione delle password degli utenti primari con Secrets Manager non è supportata per le seguenti funzionalità:

- Cluster che fanno parte di un database globale Amazon DocumentDB
- Repliche di lettura interregionali di Amazon DocumentDB

Panoramica sulla gestione delle password degli utenti primari con AWS Secrets Manager

Con AWS Secrets Manager, puoi sostituire le credenziali codificate nel tuo codice, comprese le password del database, con una chiamata API a Secrets Manager per recuperare il segreto a livello di programmazione. Per ulteriori informazioni su Secrets Manager, consulta la Guida per l'utente di AWS Secrets Manager.

Quando memorizzi i segreti del database in Secrets Manager, il tuo account comporta dei costi. AWS Per informazioni sui prezzi, consultare Prezzi di AWS Secrets Manager.

Puoi specificare che Amazon DocumentDB gestisca la password utente principale in Secrets Manager per un cluster Amazon DocumentDB quando esegui una delle seguenti operazioni:

- Creazione del cluster
- Modifica il cluster

Quando si specifica che Amazon DocumentDB gestisce la password utente principale in Secrets Manager, Amazon DocumentDB genera la password e la memorizza in Secrets Manager. Puoi interagire direttamente con il segreto per recuperare le credenziali dell'utente principale. Puoi anche specificare una chiave gestita dal cliente per crittografare il segreto o utilizzare la chiave KMS fornita da Secrets Manager.

Amazon DocumentDB gestisce le impostazioni del segreto e lo fa ruotare ogni sette giorni per impostazione predefinita. È possibile modificare alcune impostazioni, ad esempio il programma di rotazione. Se elimini un cluster che gestisce un segreto in Secrets Manager, vengono eliminati anche il segreto e i metadati associati.

Per connetterti a un cluster con le credenziali segrete, puoi recuperare il segreto da Secrets Manager. Per ulteriori informazioni, consulta Ottenere segreti da AWS Secrets Manager e connettersi a un

database SQL utilizzando JDBC con credenziali in un AWS Secrets Manager segreto nella Guida per l'utente. AWS Secrets Manager

Applicazione della gestione Amazon DocumentDB della password utente principale in AWS Secrets Manager

Puoi utilizzare le chiavi di condizione IAM per imporre la gestione di Amazon DocumentDB della password utente principale in. AWS Secrets Manager La seguente policy non consente agli utenti di creare o ripristinare istanze o cluster a meno che la password dell'utente principale non sia gestita da Amazon DocumentDB in Secrets Manager.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "rds:CreateDBCluster"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "rds:ManageMasterUserPassword": false
                }
            }
        }
    ]
}
```

Gestione della password utente principale per un cluster con Secrets Manager

Puoi configurare la gestione della password utente principale di Amazon DocumentDB in Secrets Manager eseguendo le seguenti azioni:

- Creazione di un cluster Amazon DocumentDB

- Modifica di un cluster Amazon DocumentDB

Puoi utilizzare la console Amazon DocumentDB o la AWS CLI per eseguire queste azioni.

Protezione dei dati in Amazon DocumentDB

Il modello di responsabilità AWS https://aws.amazon.com/compliance/shared-responsibility-model/ si applica alla protezione dei dati in Amazon DocumentDB. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, vedi Domande frequenti sulla privacy dei dati Termini legali della regione . Per ulteriori informazioni sulla protezione dei dati in Europa, consulta il Centro generale sulla protezione dei dati (GDPR).

Ai fini della protezione dei dati, ti consigliamo di proteggere Account AWS le credenziali e configurare i singoli utenti con Centro identità AWS IAM o AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- SSL/TLS Usalo per comunicare con AWS le risorse. È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail. Per informazioni sull'uso dei CloudTrail percorsi per acquisire AWS le attività, consulta Lavorare con i CloudTrail percorsi nella Guida per l'AWS CloudTrail utente.
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-3 per l'accesso AWS tramite un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-3.

Ti consigliamo di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include

quando lavori con Amazon DocumentDB o altro Servizi AWS utilizzando la console, l'API o gli SDK. AWS CLI AWS I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando si fornisce un URL a un server esterno, suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la richiesta al server.

Argomenti

- Client-side crittografia a livello di campo
- Crittografia dei dati di Amazon DocumentDB inattivi
- Crittografia dei dati in transito
- Gestione delle chiavi

Client-side crittografia a livello di campo

La crittografia a livello di campo (FLE) lato client di Amazon DocumentDB consente di crittografare i dati sensibili nelle applicazioni client prima che vengano trasferiti in un cluster Amazon DocumentDB. I dati sensibili rimangono crittografati quando vengono archiviati ed elaborati in un cluster e vengono decrittografati nell'applicazione client quando vengono recuperati.

Argomenti

- Nozioni di base
- Interrogazione nel FLE sul lato client

Nozioni di base

La configurazione iniziale del FLE lato client in Amazon DocumentDB è un processo in quattro fasi che include la creazione di una chiave di crittografia, l'associazione di un ruolo all'applicazione, la configurazione dell'applicazione e la definizione del funzionamento CRUD con opzioni di crittografia.

Argomenti

- Fase 1: Creazione delle chiavi di crittografia
- Fase 2: Associare un ruolo all'applicazione
- Fase 3: Configurare l'applicazione
- Fase 4: Definire un'operazione CRUD

- Esempio: file di configurazione della crittografia a livello di campo sul lato client

Fase 1: Creazione delle chiavi di crittografia

Utilizzando AWS Key Management Service, crea una chiave simmetrica gestita dal cliente che crittografa e decrittografa il campo dei dati sensibili e concedi le autorizzazioni di utilizzo IAM necessarie. AWS KMS memorizza la chiave gestita dal cliente, utilizzata per crittografare le chiavi di dati. L'archiviazione della chiave gestita dal cliente AWS KMS rafforza il livello di sicurezza. La chiave dati è la chiave secondaria, archiviata in una raccolta Amazon DocumentDB ed è necessaria per crittografare i campi sensibili prima di archiviare il documento in Amazon DocumentDB. La chiave gestita dal cliente crittografa la chiave dati, che a sua volta crittografa e decrittografa i tuoi dati. Se utilizzi un cluster globale, puoi creare una chiave multiregionale che diversi ruoli di servizio possono utilizzare in diverse regioni.

Per ulteriori informazioni su AWS Key Management Service, incluso come creare una chiave, consulta AWS Key Management Service i concetti nella Guida per gli AWS Key Management Service sviluppatori.

Fase 2: Associare un ruolo all'applicazione

Crea una policy IAM con le AWS KMS autorizzazioni appropriate. Questa policy consente alle identità IAM a cui è collegata di crittografare e decrittografare la chiave KMS specificata nel campo delle risorse. L'applicazione assume questo ruolo IAM con cui autenticarsi. AWS KMS

La policy dovrebbe essere simile a questa:

```
{ "Effect": "Allow",
"Action": ["kms:Decrypt", "kms:Encrypt"],
"Resource": "Customer Key ARN"
}
```

Fase 3: Configurare l'applicazione

Hai definito una chiave gestita dal cliente AWS KMS e creato un ruolo IAM con le autorizzazioni necessarie per accedervi. Importa i pacchetti richiesti.

```
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. Specifica `aws` come tipo di AWS KMS provider e inserisci le credenziali dell'account recuperate nel passaggio precedente.

   ```
   provider = "aws"
   kms_providers = {
       provider: {
           "accessKeyId": current_credentials.access_key,
           "secretAccessKey": current_credentials.secret_key
       }
   }
   ```

2. Specifica la chiave gestita dal cliente per crittografare la chiave dati:

   ```
   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. Configura l' MongoClient oggetto:

   ```
   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. Genera la tua chiave dati:

```
data_key_id = client_encryption.create_data_key(provider,
    customer_key,
    key_alt_name = [key_alt_name])
```

5. Recupera la tua chiave dati esistente:

```
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})
```

Fase 4: Definire un'operazione CRUD

Definisci l'operazione CRUD con opzioni di crittografia.

1. Definisci la raccolta per write/read eliminare un singolo documento:

```
coll = client.gameinfo.users
```

2. Crittografia esplicita: crittografa i campi e inserisci:

> **Note**
>
> Deve essere fornito esattamente uno tra «key_id» o «key_alt_name».

```
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"]
})
```

Esempio: file di configurazione della crittografia a livello di campo sul lato client

Nel seguente esempio, sostituisci ciascuno *placeholder* con le tue informazioni.

```
# 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()
```

Interrogazione nel FLE sul lato client

Amazon DocumentDB supporta solo le query di uguaglianza con valori crittografati FLE sul lato client.

Ad esempio, per richiedere documenti in cui il gamerscore crittografato è pari a 500, il client utilizza un metodo di crittografia esplicito per crittografare il valore della query:

```
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 }
} )
```

Crittografia dei dati di Amazon DocumentDB inattivi

> **Note**
>
> AWS KMS sta sostituendo il termine Customer Master Key (CMK) con una AWS KMS key chiave KMS. Il concetto non è cambiato. Per evitare modifiche sostanziali, AWS KMS mantiene alcune varianti di questo termine.

Puoi crittografare i dati inattivi nel tuo cluster Amazon DocumentDB specificando l'opzione di crittografia dello storage al momento della creazione del cluster. La crittografia dello storage è abilitata a livello del cluster e viene applicata a tutte le istanze, incluse l'istanza primaria e le repliche. Viene inoltre applicata al volume di archiviazione, ai dati, agli indici, ai registri, ai backup automatici e agli snapshot.

Amazon DocumentDB utilizza l'Advanced Encryption Standard (AES-256) a 256 bit per crittografare i dati utilizzando le chiavi di crittografia archiviate in (). AWS Key Management Service AWS KMS Quando si utilizza un cluster Amazon DocumentDB con la crittografia a riposo abilitata, non è necessario modificare la logica dell'applicazione o la connessione client. Amazon DocumentDB gestisce la crittografia e la decrittografia dei dati in modo trasparente, con un impatto minimo sulle prestazioni.

Amazon DocumentDB si integra AWS KMS e utilizza un metodo noto come crittografia a inviluppo per proteggere i tuoi dati. Quando un cluster Amazon DocumentDB è crittografato con un AWS KMS, Amazon DocumentDB chiede di utilizzare la tua chiave KMS AWS KMS per generare una chiave di dati in testo cifrato per crittografare il volume di archiviazione. La chiave dati cifrata viene crittografata utilizzando la chiave KMS definita dall'utente e viene archiviata insieme ai dati crittografati e ai metadati di archiviazione. Quando Amazon DocumentDB deve accedere ai tuoi dati crittografati, richiede AWS KMS di decrittografare la chiave di dati cifrata utilizzando la tua chiave KMS e memorizza nella cache la chiave di dati in testo normale in memoria per crittografare e decrittografare in modo efficiente i dati nel volume di archiviazione.

La funzione di crittografia dello storage in Amazon DocumentDB è disponibile per tutte le dimensioni di istanza supportate e in tutti i casi in cui Amazon DocumentDB è disponibile. Regioni AWS

Abilitazione della crittografia a riposo per un cluster Amazon DocumentDB

Puoi abilitare o disabilitare la crittografia a riposo su un cluster Amazon DocumentDB quando il cluster viene eseguito il provisioning utilizzando the Console di gestione AWS o the AWS Command

Line Interface ()AWS CLI. I cluster creati utilizzando la console dispongono di crittografia inattiva per impostazione predefinita. Per impostazione predefinita, nei cluster creati utilizzando la AWS CLI crittografia a riposo è disattivata. Pertanto, è necessario abilitare esplicitamente la crittografia inattiva utilizzando il parametro `--storage-encrypted`. In entrambi i casi, dopo la creazione del cluster, non è possibile modificare l'opzione di crittografia inattiva.

Amazon DocumentDB utilizza AWS KMS per recuperare e gestire le chiavi di crittografia e per definire le politiche che controllano il modo in cui queste chiavi possono essere utilizzate. Se non specifichi un identificatore di AWS KMS chiave, Amazon DocumentDB utilizza la chiave KMS del servizio AWS gestito predefinita. Amazon DocumentDB crea una chiave KMS separata per ciascuna delle tue. Regione AWS Account AWS Per ulteriori informazioni, consulta l'argomento relativo ai concetti di base di AWS Key Management Service.

Per iniziare a creare la tua chiave KMS, consulta Getting Started nella Developer Guide. AWS Key Management Service

> **Important**
>
> È necessario utilizzare una chiave KMS di crittografia simmetrica per crittografare il cluster poiché Amazon DocumentDB supporta solo chiavi KMS di crittografia simmetrica. Non utilizzare una chiave KMS asimmetrica per tentare di crittografare i dati nei tuoi cluster Amazon DocumentDB. Per ulteriori informazioni, consulta Asymmetric keys nella Developer Guide. AWS KMS AWS Key Management Service

Se Amazon DocumentDB non è più in grado di accedere alla chiave KMS per un cluster, ad esempio quando il Account AWS proprietario della chiave è sospeso, la chiave è disabilitata, la chiave è pianificata per l'eliminazione o la policy o la concessione della chiave su cui si basa Amazon DocumentDB viene rimossa, il cluster passa prima allo stato. `inaccessible-encryption-credentials-recoverable` Mentre il cluster è in questo stato, Amazon DocumentDB interrompe le istanze del cluster e non è possibile leggere o scrivere sul cluster, ma il cluster può comunque essere ripristinato se l'accesso alla chiave KMS viene ripristinato entro 7 giorni. Se l'accesso non viene ripristinato entro 7 giorni, il cluster passa allo stato di terminale. `inaccessible-encryption-credentials` Dallo stato del terminale, il cluster non è più disponibile e lo stato corrente del database non può essere ripristinato: è possibile eseguire il ripristino solo da un backup o eseguire un ripristino point-in-time utilizzando la chiave KMS originale. Per Amazon DocumentDB, i backup sono sempre abilitati per almeno 1 giorno.

> **Note**
>
> I cluster che fanno parte di un cluster globale si comportano diversamente. Quando Amazon DocumentDB rileva che non può più accedere alla chiave KMS, tutti i cluster del cluster globale passano direttamente allo stato di terminale, saltando `inaccessible-encryption-credentials` lo stato ripristinabile. Questo perché un cluster che fa parte di un cluster globale può essere interrotto e avviato solo quando è l'unico cluster nel cluster globale. Per eseguire il ripristino, è necessario eseguire il ripristino da un'istantanea o eseguire un ripristino point-in-time. Per eliminare i cluster originali, è necessario prima rimuovere ogni cluster dal cluster globale e quindi eliminarlo.

> **Important**
>
> Non è possibile modificare la chiave KMS per un cluster crittografato dopo averlo già creato. Assicurati di determinare i requisiti della chiave crittografica prima di creare il tuo cluster crittografato.

Using the Console di gestione AWS

Puoi specificare l'opzione crittografia inattiva al momento della creazione di un cluster. La crittografia inattiva è abilitata per impostazione predefinita quando si crea un cluster utilizzando l'opzione Console di gestione AWS. Non può essere modificata dopo la creazione del cluster.

Per specificare l'opzione crittografia inattiva durante la creazione del cluster

1. Crea un cluster Amazon DocumentDB come descritto nella sezione Guida introduttiva. Tuttavia, nel passaggio 6, non scegliere Create cluster (Crea cluster).
2. Nella sezione Authentication (Autenticazione), scegliere Show advanced settings (Mostra impostazioni avanzate).
3. Scorri verso il basso fino alla Encryption-at-rest sezione.
4. Scegliere l'opzione desiderata per la crittografia inattiva. Qualunque sia l'opzione scelta, non è possibile modificarla dopo la creazione del cluster.
   - Per crittografare i dati inattivi in questo cluster, scegliere Enable encryption (Abilita crittografia).

- Se non si desidera crittografare i dati inattivi in questo cluster, scegliere Disable encryption (Disabilita crittografia).

5. Scegli la chiave primaria che desideri. Amazon DocumentDB utilizza AWS Key Management Service (AWS KMS) per recuperare e gestire le chiavi di crittografia e per definire le politiche che controllano il modo in cui queste chiavi possono essere utilizzate. Se non specifichi un identificatore di AWS KMS chiave, Amazon DocumentDB utilizza la chiave KMS del servizio AWS gestito predefinita. Per ulteriori informazioni, consulta l'argomento relativo ai concetti di base di AWS Key Management Service.

   > **Note**
   >
   > Dopo aver creato un cluster crittografato, non puoi modificare la chiave KMS per quel cluster. Assicurati di determinare i requisiti della chiave crittografica prima di creare il tuo cluster crittografato.

6. Completare le altre sezioni secondo necessità e creare il cluster.

Using the AWS CLI

Per crittografare un cluster Amazon DocumentDB utilizzando AWS CLI, esegui il `create-db-cluster` comando e specifica l'opzione. `--storage-encrypted` I cluster Amazon DocumentDB creati utilizzando il AWS CLI non abilitano la crittografia dello storage per impostazione predefinita.

L'esempio seguente crea un cluster Amazon DocumentDB con la crittografia dello storage abilitata.

Nei seguenti esempi, sostituisci ciascuno *`user input placeholder`* con le informazioni del tuo cluster.

Example

Per Linux, macOS o Unix:

```
aws docdb create-db-cluster \
  --db-cluster-identifier mydocdbcluster \
  --port 27017 \
  --engine docdb \
```

```
  --master-username SampleUser1 \
  --master-user-password primaryPassword \
  --storage-encrypted
```

Per Windows:

```
aws docdb create-db-cluster ^
  --db-cluster-identifier SampleUser1 ^
  --port 27017 ^
  --engine docdb ^
  --master-username SampleUser1 ^
  --master-user-password primaryPassword ^
  --storage-encrypted
```

Quando crei un cluster Amazon DocumentDB crittografato, puoi specificare un identificatore AWS KMS chiave, come nell'esempio seguente.

Example

Per Linux, macOS o 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
```

Per 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**
>
> Dopo aver creato un cluster crittografato, non puoi modificare la chiave KMS per quel cluster. Assicurati di determinare i requisiti della chiave crittografica prima di creare il tuo cluster crittografato.

Risoluzione di un cluster Amazon DocumentDB in uno stato di crittografia inaccessibile

Un cluster Amazon DocumentDB passa a uno stato di crittografia inaccessibile quando Amazon DocumentDB non può accedere alla chiave KMS con cui il cluster è stato crittografato. Esistono due stati di questo tipo e il percorso di ripristino dipende da quello in cui si trova il cluster.

`stato` inaccessible-encryption-credentials-recoverable

Mentre `inaccessible-encryption-credentials-recoverable` lo stato del cluster è attivo, puoi ripristinarlo ripristinando l'accesso di Amazon DocumentDB alla chiave KMS e quindi `available` avviando il cluster. Per risolvere questo stato, procedi come segue:

1. Verifica Account AWS che il proprietario della chiave KMS sia attivo. Se l'account è sospeso, riattivalo.
2. Conferma che la chiave KMS sia abilitata. Per ulteriori informazioni, consulta Abilitazione e disabilitazione delle chiavi nella Guida per gli AWS Key Management Service sviluppatori.
3. Verifica se è prevista l'eliminazione della chiave KMS. In caso affermativo, annulla l'eliminazione pianificata della chiave. Per ulteriori informazioni, consulta Pianificazione e annullamento dell'eliminazione delle chiavi nella Developer Guide. AWS Key Management Service
4. Verifica che la politica delle chiavi KMS e le eventuali concessioni su cui si basa Amazon DocumentDB consentano comunque ad Amazon DocumentDB di utilizzare la chiave.
5. Dopo aver ripristinato l'accesso alla chiave KMS, avvia il cluster utilizzando o eseguendo il Console di gestione AWS comando. `start-db-cluster` AWS CLI

   Example

   Per Linux, macOS o Unix:

   ```
   aws docdb start-db-cluster \
     --db-cluster-identifier example-cluster
   ```

Per Windows:

```
aws docdb start-db-cluster ^
  --db-cluster-identifier example-cluster
```

> **Important**
>
> Se l'accesso alla chiave KMS non viene ripristinato entro 7 giorni, il cluster passa allo `inaccessible-encryption-credentials` stato di terminale, da cui non può essere avviato.

`stato delle credenziali di crittografia inaccessibili`

Lo stato è terminale. `inaccessible-encryption-credentials` Il cluster non può essere avviato e lo stato di esecuzione del database non può essere ripristinato. Per ripristinare i dati, esegui il ripristino da un'istantanea o esegui un ripristino point-in-time su un nuovo cluster. È necessario avere ancora accesso alla chiave KMS originale per eseguire il ripristino. Se la chiave KMS è stata eliminata, i dati non possono essere recuperati.

Per ulteriori informazioni, consulta Ripristino da un'istantanea del cluster e Ripristino a un determinato punto nel tempo.

> **Note**
>
> I cluster che fanno parte di un cluster globale passano direttamente allo `inaccessible-encryption-credentials` stato in cui si perde l'accesso alla chiave KMS, poiché un cluster che fa parte di un cluster globale può essere interrotto e avviato solo quando è l'unico cluster nel cluster globale. Per eliminare un cluster che si trova in questo stato, rimuovi prima ogni cluster dal cluster globale, quindi elimina i cluster singolarmente.

Se non riesci a eliminare un cluster che si trova in tale `inaccessible-encryption-credentials` stato perché la protezione dall'eliminazione è abilitata, disattiva la protezione dall'eliminazione utilizzando il AWS CLI prima di riprovare l'eliminazione.

Example

Per Linux, macOS o Unix:

```
aws docdb modify-db-cluster \
  --db-cluster-identifier example-cluster \
  --no-deletion-protection
```

Per Windows:

```
aws docdb modify-db-cluster ^
  --db-cluster-identifier example-cluster ^
  --no-deletion-protection
```

È quindi possibile eliminare il cluster utilizzando il comando. `delete-db-cluster`

Example

Per Linux, macOS o Unix:

```
aws docdb delete-db-cluster \
  --db-cluster-identifier example-cluster \
  --skip-final-snapshot
```

Per Windows:

```
aws docdb delete-db-cluster ^
  --db-cluster-identifier example-cluster ^
  --skip-final-snapshot
```

Se il cluster non viene eliminato dopo aver eseguito i comandi precedenti, contatta Supporto AWS.

Limitazioni per i cluster crittografati di Amazon DocumentDB

Esistono le seguenti limitazioni per i cluster crittografati di Amazon DocumentDB.

- È possibile abilitare o disabilitare la crittografia a riposo per un cluster Amazon DocumentDB solo al momento della creazione, non dopo la creazione del cluster. Tuttavia, puoi creare una copia crittografata di un cluster non crittografato creando un'istantanea del cluster non crittografato e quindi ripristinando l'istantanea non crittografata come nuovo cluster specificando l'opzione di crittografia a riposo.

Per ulteriori informazioni, consulta i seguenti argomenti:

- Creazione manuale di un'istantanea del cluster
- Ripristino da un'istantanea del cluster
- Copia delle istantanee del cluster Amazon DocumentDB

- I cluster Amazon DocumentDB con crittografia dello storage abilitata non possono essere modificati per disabilitare la crittografia.
- Tutte le istanze, i backup automatici, le istantanee e gli indici in un cluster Amazon DocumentDB sono crittografati con la stessa chiave KMS.

Crittografia dei dati in transito

Puoi utilizzare Transport Layer Security (TLS) per crittografare la connessione tra la tua applicazione e un cluster Amazon DocumentDB. Per impostazione predefinita, la crittografia in transito è abilitata per i cluster Amazon DocumentDB appena creati. Può facoltativamente essere disabilitata al momento della creazione del cluster o in un secondo momento. Quando la crittografia in transito è abilitata, per connettersi al cluster sono necessarie connessioni protette tramite TLS. Per ulteriori informazioni sulla connessione ad Amazon DocumentDB utilizzando TLS, consulta Connessione programmatica ad Amazon DocumentDB.

Gestione delle impostazioni TLS del cluster Amazon DocumentDB

La crittografia in transito per un cluster Amazon DocumentDB viene gestita tramite il parametro TLS in un gruppo di parametri del cluster. Puoi gestire le impostazioni TLS del cluster Amazon DocumentDB utilizzando Console di gestione AWS o (). AWS Command Line Interface AWS CLI Consulta le seguenti sezioni per ulteriori informazioni su come verificare e modificare le impostazioni TLS correnti.

Using the Console di gestione AWS

Segui questi passaggi per eseguire attività di gestione della crittografia TLS utilizzando la console, ad esempio identificare i gruppi di parametri, verificare il valore TLS e apportare le modifiche necessarie.

> **Note**
>
> A meno che non si specifichi diversamente quando si crea un cluster, il cluster viene creato con il gruppo di parametri cluster predefinito. I parametri nel gruppo di parametri

del `default` cluster non possono essere modificati (ad esempio,). `tls` enabled/disabled Pertanto, se il cluster utilizza un gruppo di parametri del cluster `default`, è necessario modificare il cluster per utilizzare un gruppo di parametri cluster non predefinito. Innanzitutto, potrebbe essere necessario creare un gruppo di parametri cluster personalizzato. Per ulteriori informazioni, consulta Creazione di gruppi di parametri del cluster Amazon DocumentDB.

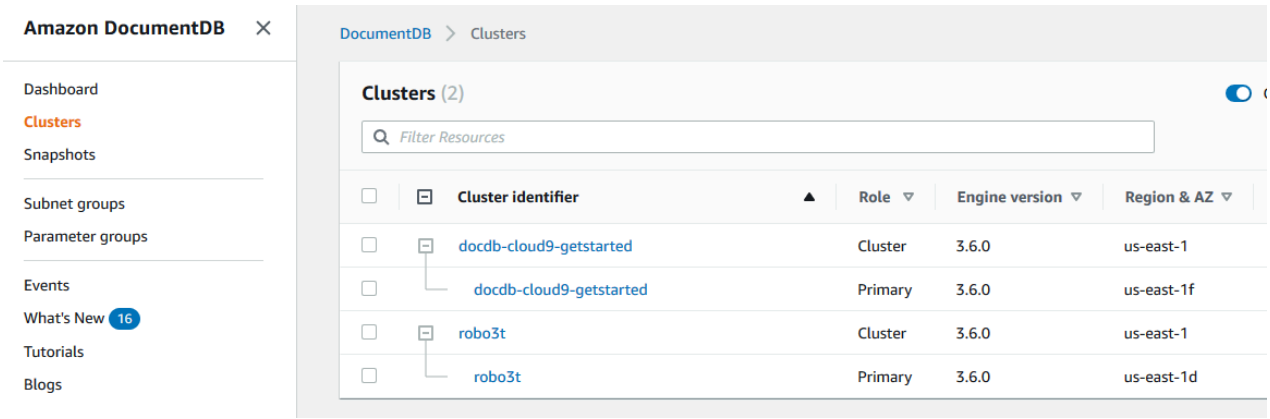
1. Determinare il gruppo di parametri cluster utilizzato dal cluster.

   a. Apri la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.

   b. Nel pannello di navigazione scegliere Cluster.

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

   c. Nota che nella casella di navigazione Clusters, la colonna Cluster Identifier mostra sia i cluster che le istanze. Le istanze sono elencate sotto i cluster. Guarda la schermata seguente per riferimento.



   d. Scegli il cluster che ti interessa.

   e. Scegli la scheda Configurazione e scorri fino alla fine dei dettagli del cluster e individua il gruppo di parametri Cluster. Annotare il nome del gruppo di parametri del cluster.

Se il nome del gruppo di parametri del cluster è `default`, ad esempio `default.docdb3.6`, è necessario disporre di un gruppo di parametri del cluster personalizzato e renderlo il gruppo di parametri del cluster prima di continuare. Per ulteriori informazioni, consulta gli argomenti seguenti:

1. Creazione di gruppi di parametri del cluster Amazon DocumentDB— Se non disponi di un gruppo di parametri del cluster personalizzato da utilizzare, creane uno.
2. Modifica di un cluster Amazon DocumentDB— Modifica il cluster per utilizzare il gruppo di parametri del cluster personalizzato.

2. Determinare l'attuale valore del parametro cluster **`tls`**.

   a. Apri la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.

   b. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   c. Dall'elenco dei gruppi di parametri del cluster, scegliere il nome del gruppo che ti interessa.

   d. Individuare la sezione Cluster parameters (Parametri cluster). Nell'elenco dei parametri del cluster individuare la riga dei parametri del cluster `tls`. A questo punto, sono importanti le seguenti quattro colonne:

      - Nome del parametro del cluster: il nome dei parametri del cluster. Per gestire TLS, ti interessa il parametro `tls` del cluster.
      - Valori: il valore corrente di ogni parametro del cluster.
      - Valori consentiti: un elenco di valori che possono essere applicati a un parametro del cluster.
      - Tipo di applicazione: statico o dinamico. Le modifiche apportate ai parametri del cluster statico possono essere applicate solo quando le istanze vengono riavviate. Le modifiche apportate ai parametri del cluster dinamico possono essere applicate immediatamente oppure quando le istanze vengono riavviate.

3. Modificare il valore del parametro del cluster **`tls`**.

   Se il valore di `tls` non corrisponde a quello richiesto, modificare il suo valore per questo gruppo di parametri del cluster. Per modificare il valore del parametro del cluster `tls`, continuare dalla sezione precedente seguendo questi passaggi.

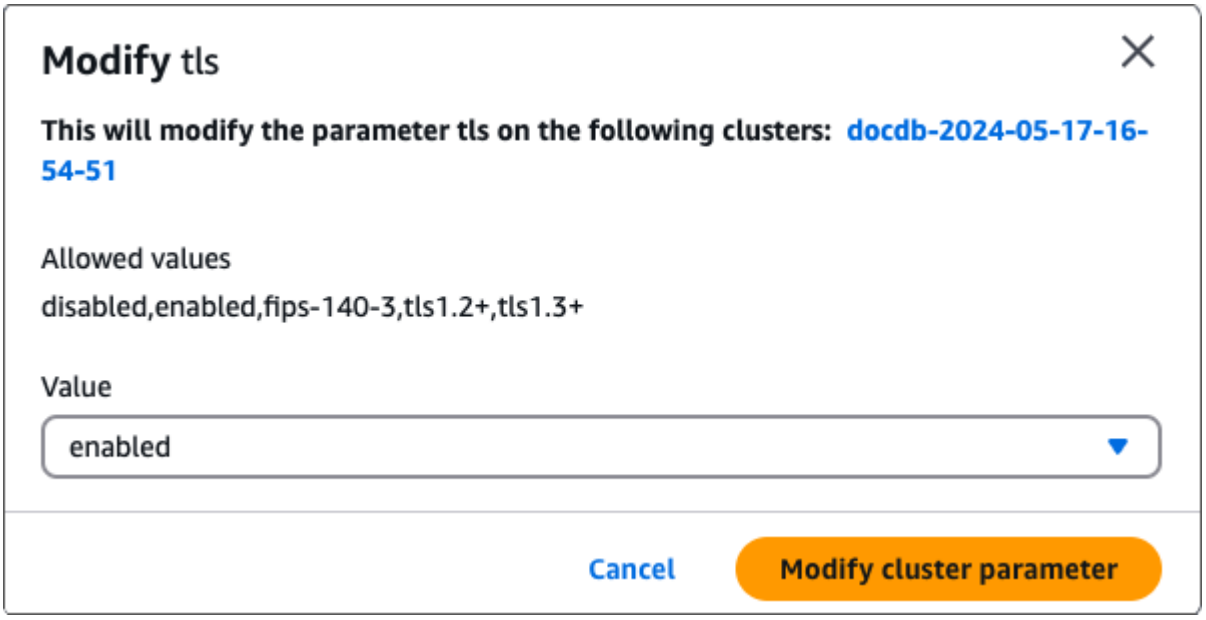
   a. Scegliere il pulsante a sinistra del nome del parametro cluster (`tls`).

b. Scegli Modifica.

c. Per modificare il valore di`tls`, nella finestra di `tls` dialogo Modifica, scegli il valore che desideri per il parametro cluster nell'elenco a discesa.

   I valori validi sono:

   - disabilitato: disabilita TLS
   - abilitato: abilita TLS. Per Amazon DocumentDB 5.0 e versioni precedenti, abilita TLS da 1.0 a 1.3, ma utilizza TLS 1.2 o versioni successive. Solo TLS 1.2 e versioni successive sono supportati su Amazon DocumentDB 5.0 a partire dalla versione secondaria 5.0.1, Amazon DocumentDB 8.0 e successive.
   - fips-140-3: abilita TLS con FIPS. Il cluster accetta solo connessioni sicure in base ai requisiti della pubblicazione 140-3 degli Federal Information Processing Standards (FIPS). È supportato solo a partire dai cluster Amazon DocumentDB 5.0 (versione del motore 3.0.3727) nelle seguenti regioni: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1.
   - tls1.2+: abilita TLS versione 1.2 e successive. È supportato solo a partire da Amazon DocumentDB 4.0 (versione del motore 2.0.10980) e Amazon DocumentDB (versione del motore 3.0.11051).
   - tls1.3+: abilita la versione TLS 1.3 e successive. È supportato solo a partire da Amazon DocumentDB 4.0 (versione del motore 2.0.10980) e Amazon DocumentDB (versione del motore 3.0.11051).



d. Scegliere Modify cluster parameter (Modifica parametro cluster). La modifica verrà applicata a ciascuna istanza di cluster quando viene riavviata.

4. Riavvia l'istanza Amazon DocumentDB.

   Riavviare ogni istanza del cluster in modo che la modifica venga applicata a tutte le istanze nel cluster.

   a. Apri la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.

   b. Nel riquadro di navigazione, scegliere Instances (Istanze).

   c. Per specificare un'istanza da riavviare, trova l'istanza nell'elenco e scegli il pulsante a sinistra del nome.

   d. Scegliere Actions (Azioni), quindi Reboot (Riavvia). Confermare che si desidera riavviare scegliendo Reboot (Riavvia).

Using the AWS CLI

Segui questi passaggi per eseguire attività di gestione della crittografia TLS utilizzando, AWS CLI ad esempio, l'identificazione dei gruppi di parametri, la verifica del valore TLS e l'apporto delle modifiche necessarie.

> **Note**
>
> A meno che non si specifichi diversamente quando si crea un cluster, il cluster viene creato con il gruppo di parametri cluster predefinito. I parametri nel gruppo di parametri del `default` cluster non possono essere modificati (ad esempio,). `tls` enabled/disabled Pertanto, se il cluster utilizza un gruppo di parametri del cluster `default`, è necessario modificare il cluster per utilizzare un gruppo di parametri cluster non predefinito. Potrebbe essere necessario creare prima un gruppo di parametri del cluster personalizzato. Per ulteriori informazioni, consulta Creazione di gruppi di parametri del cluster Amazon DocumentDB.

1. Determinare il gruppo di parametri cluster utilizzato dal cluster.

   Esegui il `describe-db-clusters` comando con le seguenti opzioni:

   - `--db-cluster-identifier`
   - `--query`

   Nel seguente esempio, sostituisci ciascuno *`user input placeholder`* con le informazioni del tuo cluster.

   ```
   aws docdb describe-db-clusters \
     --db-cluster-identifier mydocdbcluster \
     --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
   ```

   L'output di questa operazione è simile al seguente (formato JSON):

   ```
   [
       [
           "mydocdbcluster",
           "myparametergroup"
       ]
   ]
   ```

   Se il nome del gruppo di parametri del cluster è `default`, ad esempio `default.docdb3.6`, è necessario disporre di un gruppo di parametri del cluster personalizzato e renderlo il gruppo

di parametri del cluster prima di continuare. Per ulteriori informazioni, consulta i seguenti argomenti:

1. Creazione di gruppi di parametri del cluster Amazon DocumentDB— Se non disponi di un gruppo di parametri del cluster personalizzato da utilizzare, creane uno.
2. Modifica di un cluster Amazon DocumentDB— Modifica il cluster per utilizzare il gruppo di parametri del cluster personalizzato.

2. Determinare l'attuale valore del parametro cluster **`tls`**.

Per ottenere ulteriori informazioni su questo gruppo di parametri del cluster, esegui il `describe-db-cluster-parameters` comando con le seguenti opzioni:

- `--db-cluster-parameter-group-name`
- `--query`

  Limita l'output ai soli campi di interesse: `ParameterNameParameterValue`,`AllowedValues`, `eApplyType`.

Nell'esempio seguente, sostituisci ciascuno *`user input placeholder`* con le informazioni del tuo cluster.

```
aws docdb describe-db-cluster-parameters \
  --db-cluster-parameter-group-name myparametergroup \
  --query 'Parameters[*].[ParameterName,ParameterValue,AllowedValues,ApplyType]'
```

L'output di questa operazione è simile al seguente (formato JSON):

```
[
    [
        "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. Modificare il valore del parametro del cluster **`tls`**.

   Se il valore di `tls` non corrisponde a quello richiesto, modificarlo per questo gruppo di parametri del cluster. Per modificare il valore del parametro `tls` cluster, esegui il [modify-db-cluster-parameter-group](#) comando con le seguenti opzioni:

   - `--db-cluster-parameter-group-name`: obbligatorio. Il nome del gruppo di parametri del cluster da modificare. Questo non può essere un gruppo di parametri del `default.*`.
   - `--parameters`: obbligatorio. Un elenco di parametri del gruppo di parametri del cluster.
     - `ParameterName`: obbligatorio. Il nome del parametro del cluster da modificare.
     - `ParameterValue`: obbligatorio. Il nuovo valore per questo parametro del cluster. Deve essere uno dei valori `AllowedValues` del parametro del cluster.
       - `enabled`— Il cluster accetta connessioni sicure tramite TLS. Per Amazon DocumentDB 5.0 e versioni precedenti, questo abilita TLS da 1.0 a 1.3, ma utilizza TLS 1.2 o versioni successive. Solo TLS 1.2 e versioni successive sono supportati su Amazon DocumentDB 5.0 a partire dalla versione secondaria 5.0.1, Amazon DocumentDB 8.0 e successive.
       - `disabled`— Il cluster non accetta connessioni sicure tramite TLS.
       - `fips-140-3`— Il cluster accetta solo connessioni sicure secondo i requisiti della pubblicazione 140-3 degli Federal Information Processing Standards (FIPS). È supportato solo a partire dai cluster Amazon DocumentDB 5.0 (versione del motore 3.0.3727) nelle seguenti regioni: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1.
       - `tls1.2+`— Il cluster accetta connessioni sicure utilizzando TLS versione 1.2 e successive. È supportato solo a partire da Amazon DocumentDB 4.0 (versione del motore 2.0.10980) e Amazon DocumentDB 5.0 (versione del motore 3.0.11051).

    - `tls1.3+`— Il cluster accetta connessioni sicure utilizzando TLS versione 1.3 e successive. È supportato solo a partire da Amazon DocumentDB 4.0 (versione del motore 2.0.10980) e Amazon DocumentDB 5.0 (versione del motore 3.0.11051).
- `ApplyMethod`— Quando deve essere applicata questa modifica. Per i parametri del cluster statici, ad esempio `tle`, questo valore deve essere `pending-reboot`.
    - `pending-reboot`— La modifica viene applicata a un'istanza solo dopo il riavvio. È necessario riavviare ogni istanza del cluster individualmente per consentire l'applicazione di questo cambiamento tra tutte le istanze del cluster.

Negli esempi seguenti, sostituisci ciascuno *`user input placeholder`* con le informazioni del tuo cluster.

Il codice seguente disabilita `tls`, applicando la modifica a ogni istanza al riavvio.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=tls,ParameterValue=disabled,ApplyMethod=pending-
reboot"
```

Il codice seguente abilita `tls`, applicando la modifica a ogni istanza al riavvio.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=tls,ParameterValue=enabled,ApplyMethod=pending-
reboot"
```

Il codice seguente abilita TLS con`fips-140-3`, applicando la modifica a ciascuna istanza al riavvio.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup2 \
  --parameters "ParameterName=tls,ParameterValue=fips-140-3,ApplyMethod=pending-
reboot"
```

L'output di questa operazione è simile al seguente (formato JSON):

```
{
    "DBClusterParameterGroupName": "myparametergroup"
```

```
}
```

4. Riavvia la tua istanza Amazon DocumentDB.

   Riavviare ogni istanza del cluster in modo che la modifica venga applicata a tutte le istanze nel cluster. Per riavviare un'istanza Amazon DocumentDB, esegui il `reboot-db-instance` comando con la seguente opzione:

   - `--db-instance-identifier`

   Il codice seguente riavvia l'istanza `mydocdbinstance`.

   Nei seguenti esempi, sostituisci ciascuno *`user input placeholder`* con le informazioni del tuo cluster.

   Example

   Per Linux, macOS o Unix:

```
aws docdb reboot-db-instance \
  --db-instance-identifier mydocdbinstance
```

   Per Windows:

```
aws docdb reboot-db-instance ^
  --db-instance-identifier mydocdbinstance
```

   L'output di questa operazione è simile al seguente (formato JSON):

```
{
    "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"
    }
}
```

Per il riavvio dell'istanza sono necessari alcuni minuti. Puoi utilizzare l'istanza solo quando ha lo stato disponibile. Puoi monitorare lo stato dell'istanza con la console o l' AWS CLI. Per ulteriori informazioni, consulta Monitoraggio dello stato di un'istanza Amazon DocumentDB.

Gestione delle chiavi

Amazon DocumentDB utilizza AWS Key Management Service (AWS KMS) per recuperare e gestire le chiavi di crittografia. AWS KMS combina hardware e software sicuri e ad alta disponibilità per fornire un sistema di gestione delle chiavi scalabile per il cloud. Utilizzando AWS KMS, è possibile

creare chiavi di crittografia e definire le politiche che controllano il modo in cui queste chiavi possono essere utilizzate. AWS KMS supporta AWS CloudTrail, in modo da poter controllare l'utilizzo delle chiavi per verificare che le chiavi vengano utilizzate in modo appropriato.

Puoi utilizzare le tue chiavi KMS in combinazione con Amazon DocumentDB e AWS servizi supportati come Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), Amazon Elastic Block Store (Amazon EBS) e Amazon Redshift. Per un elenco dei servizi che supportano AWS KMS, consulta How AWS services use nella Developer Guide. AWS KMS AWS Key Management Service Per informazioni su AWS KMS, vedi Che cos'è AWS Key Management Service?

Gestione delle identità e degli accessi per Amazon DocumentDB

AWS Identity and Access Management (IAM) è una soluzione Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle risorse. AWS Gli amministratori IAM controllano chi può essere autenticato (accedere) e autorizzato (disporre delle autorizzazioni) a utilizzare le risorse Amazon DocumentDB. IAM è un programma Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Argomenti

- Destinatari
- Autenticazione con identità
- Gestione dell’accesso tramite policy
- Come funziona Amazon DocumentDB con IAM
- Identity-based esempi di policy per Amazon DocumentDB
- Risoluzione dei problemi di identità e accesso ad Amazon DocumentDB
- Gestione delle autorizzazioni di accesso alle tue risorse Amazon DocumentDB
- Utilizzo di policy basate sull'identità (politiche IAM) per Amazon DocumentDB
- AWS politiche gestite per Amazon DocumentDB
- Autorizzazioni API Amazon DocumentDB: riferimento ad azioni, risorse e condizioni

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia in base al tuo ruolo:

- Utente del servizio: richiedi le autorizzazioni all'amministratore se non riesci ad accedere alle funzionalità (consulta Risoluzione dei problemi di identità e accesso ad Amazon DocumentDB)
- Amministratore del servizio: determina l'accesso degli utenti e invia le richieste di autorizzazione (consulta Come funziona Amazon DocumentDB con IAM)
- Amministratore IAM: scrivi policy per gestire l'accesso (consulta Identity-based esempi di policy per Amazon DocumentDB)

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi essere autenticato come utente IAM o assumendo un ruolo IAM. Utente root dell'account AWS

Puoi accedere come identità federata utilizzando credenziali da una fonte di identità come Centro identità AWS IAM (IAM Identity Center), autenticazione Single Sign-on o credenziali. Google/ Facebook Per ulteriori informazioni sull'accesso, consulta Come accedere all' Account AWS nella Guida per l'utente di Accedi ad AWS .

Per l'accesso programmatico, AWS fornisce un SDK e una CLI per firmare crittograficamente le richieste. Per ulteriori informazioni, consulta AWS Signature Version 4 per le richieste API nella Guida per l'utente di IAM.

Account AWS utente root

Quando ne crei uno Account AWS, inizi con un'identità di accesso chiamata utente Account AWS root che ha accesso completo a tutte Servizi AWS le risorse. Consigliamo vivamente di non utilizzare l'utente root per le attività quotidiane. Per le attività che richiedono le credenziali dell'utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'utente IAM.

Identità federata

È consigliabile richiedere agli utenti umani di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente della directory aziendale, del provider di identità web o Directory Service che accede Servizi AWS utilizzando credenziali da una fonte di identità. Le identità federate assumono ruoli che forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, si consiglia di utilizzare Centro identità AWS IAM. Per ulteriori informazioni, consulta Che cos'è il Centro identità IAM? nella Guida per l'utente di Centro identità AWS IAM .

Utenti e gruppi IAM

Un utente IAM è una identità che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ti consigliamo di utilizzare credenziali temporanee invece di utenti IAM con credenziali a lungo termine. Per ulteriori informazioni, consulta Richiedere agli utenti umani di utilizzare la federazione con un provider di identità per accedere AWS utilizzando credenziali temporanee nella IAM User Guide.

Un gruppo IAM specifica una raccolta di utenti IAM e semplifica la gestione delle autorizzazioni per gestire gruppi di utenti di grandi dimensioni. Per ulteriori informazioni, consulta Casi d'uso per utenti IAM nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità con autorizzazioni specifiche che fornisce credenziali temporanee. Puoi assumere un ruolo passando da un utente a un ruolo IAM (console) o chiamando un'operazione AWS API AWS CLI or. Per ulteriori informazioni, consulta Metodi per assumere un ruolo nella Guida per l'utente di IAM.

I ruoli IAM sono utili per l'accesso degli utenti federati, le autorizzazioni utente IAM temporanee, l'accesso multi-account, l'accesso multi-servizio e le applicazioni in esecuzione su Amazon EC2. Per maggiori informazioni, consultare Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Gestione dell'accesso tramite policy

Puoi controllare l'accesso AWS creando policy e associandole a AWS identità o risorse. Una policy definisce le autorizzazioni quando è associata a un'identità o a una risorsa. AWS valuta queste politiche quando un principale effettua una richiesta. La maggior parte delle policy sono archiviate AWS come documenti JSON. Per maggiori informazioni sui documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente IAM.

Utilizzando le policy, gli amministratori specificano chi ha accesso a cosa definendo quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Un amministratore IAM crea le policy IAM e le aggiunge ai ruoli, che gli utenti possono quindi assumere. Le policy IAM definiscono le autorizzazioni indipendentemente dal metodo utilizzato per eseguirle.

Identity-based politiche

Identity-based le policy sono documenti relativi alle politiche di autorizzazione JSON che si allegano a un'identità (utente, gruppo o ruolo). Tali policy controllano le operazioni autorizzate per l'identità, nonché le risorse e le condizioni in cui possono essere eseguite. Per informazioni su come creare una policy basata su identità, consultare Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente IAM.

Identity-based le politiche possono essere politiche in linea (incorporate direttamente in una singola identità) o politiche gestite (politiche autonome associate a più identità). Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scegliere tra policy gestite e policy in linea nella Guida per l'utente di IAM.

Resource-based politiche

Resource-based le policy sono documenti di policy JSON che alleghi a una risorsa. Gli esempi includono le policy di trust dei ruoli IAM e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. In una policy basata sulle risorse è obbligatorio specificare un'entità principale.

Resource-based le policy sono policy in linea che si trovano in quel servizio. Non è possibile utilizzare le policy AWS gestite da IAM in una policy basata sulle risorse.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi che possono impostare le autorizzazioni massime concesse dai tipi di policy più comuni:

- Limiti delle autorizzazioni: imposta il numero massimo di autorizzazioni che una policy basata su identità ha la possibilità di concedere a un'entità IAM. Per ulteriori informazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Policy di controllo dei servizi (SCP): specifica il numero massimo di autorizzazioni per un'organizzazione o un'unità organizzativa (OU) in AWS Organizations. Per ulteriori informazioni, consultare Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations .

- Policy di controllo delle risorse (RCP): imposta le autorizzazioni massime disponibili per le risorse degli account. Per ulteriori informazioni, consulta Policy di controllo delle risorse (RCP) nella Guida per l'utente di AWS Organizations .
- Policy di sessione: policy avanzate passate come parametro quando si crea una sessione temporanea per un ruolo o un utente federato. Per maggiori informazioni, consultare Policy di sessione nella Guida per l'utente IAM.

Più tipi di policy

Quando a una richiesta si applicano più tipi di policy, le autorizzazioni risultanti sono più complicate da comprendere. Per sapere come AWS determinare se consentire una richiesta quando sono coinvolti più tipi di policy, consulta la logica di valutazione delle policy nella IAM User Guide.

Come funziona Amazon DocumentDB con IAM

Prima di utilizzare IAM per gestire l'accesso ad Amazon DocumentDB, scopri quali funzionalità IAM sono disponibili per l'uso con Amazon DocumentDB.

Funzionalità IAM che puoi usare con Amazon DocumentDB

Funzionalità IAM	Instance-based cluster	Cluster elastici
Identity-based politiche	Sì	Sì
Resource-based politiche	No	No
Operazioni di policy	Sì	Sì
Risorse relative alle policy	Sì	Sì
Chiavi di condizione della policy (specifica del servizio)	Sì	Sì
Liste di controllo degli accessi (ACL)	No	No
ABAC (tag nelle policy)	Parziale	Sì
Credenziali temporanee	Sì	Sì

Funzionalità IAM	Instance-based cluster	Cluster elastici
Autorizzazioni del principale	Sì	Sì
Ruoli di servizio	Sì	Sì
Service-linked ruoli	No	Sì

Per avere una visione generale di come Amazon DocumentDB e altri AWS servizi funzionano con la maggior parte delle funzionalità IAM, consulta AWS i servizi che funzionano con IAM nella IAM User Guide.

Identity-based politiche per Amazon DocumentDB

Supporta le policy basate sull'identità: sì

Identity-based le policy sono documenti di policy sulle autorizzazioni JSON che puoi allegare a un'identità, ad esempio un utente, un gruppo di utenti o un ruolo IAM. Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente di IAM.

Con le policy basate sull'identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Per informazioni su tutti gli elementi utilizzabili in una policy JSON, consulta Guida di riferimento agli elementi delle policy JSON IAM nella Guida per l'utente IAM.

Identity-based esempi di policy per Amazon DocumentDB

Per visualizzare esempi di politiche basate sull'identità di Amazon DocumentDB, consulta. Identity-based esempi di policy per Amazon DocumentDB

Resource-based politiche all'interno di Amazon DocumentDB

Supporta le policy basate su risorse: no

Resource-based le policy sono documenti di policy JSON che alleghi a una risorsa. Esempi di policy basate sulle risorse sono le policy di attendibilità dei ruoli IAM e le policy di bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per

controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le operazioni che un principale può eseguire su tale risorsa e a quali condizioni. In una policy basata sulle risorse è obbligatorio specificare un'entità principale. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Per consentire l'accesso multi-account, è possibile specificare un intero account o entità IAM in un altro account come entità principale in una policy basata sulle risorse. Per ulteriori informazioni, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Azioni politiche per Amazon DocumentDB

Supporta le operazioni di policy: si

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso in una policy. Includere le operazioni in una policy per concedere le autorizzazioni di eseguire l'operazione associata.

> **Note**
>
> Per alcune funzionalità di gestione, Amazon DocumentDB utilizza la tecnologia operativa condivisa con Amazon Relational Database Service (Amazon RDS).
> Per visualizzare un elenco di azioni RDS, consulta Azioni definite da Amazon Relational Database Service nel Service Authorization Reference.
> Per visualizzare le azioni politiche per i cluster elastici di Amazon DocumentDB, consulta Azioni definite dai cluster elastici Amazon DocumentDB nel Service Authorization Reference.

Le azioni politiche in Amazon DocumentDB utilizzano il seguente prefisso prima dell'azione:

```
aws
```

Per specificare più operazioni in una sola istruzione, occorre separarle con la virgola.

```
"Action": [
      "aws:action1",
      "aws:action2"
```

```
        ]
```

Per visualizzare esempi di policy basate sull'identità di Amazon DocumentDB, consulta. Identity-based esempi di policy per Amazon DocumentDB

Risorse relative alle policy per Amazon DocumentDB

Supporta le risorse relative alle policy: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). Per le azioni che non supportano le autorizzazioni a livello di risorsa, si utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

> **Note**
>
> Per alcune funzionalità di gestione, Amazon DocumentDB utilizza la tecnologia operativa condivisa con Amazon Relational Database Service (Amazon RDS).
> Per visualizzare un elenco dei tipi di risorse RDS e dei relativi ARN, consulta Resources defined by Amazon Relational Database Service nel Service Authorization Reference. Per sapere con quali azioni è possibile specificare l'ARN di ciascuna risorsa, consulta Azioni definite da Amazon Relational Database Service.
> Per visualizzare i tipi di risorse per i cluster elastici di Amazon DocumentDB, consulta Tipi di risorse definiti dai cluster elastici Amazon DocumentDB nel Service Authorization Reference.

Per visualizzare esempi di policy basate sull'identità di Amazon DocumentDB, consulta. Identity-based esempi di policy per Amazon DocumentDB

Chiavi delle condizioni delle policy per Amazon DocumentDB

Supporta le chiavi di condizione delle policy specifiche del servizio: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Condition` specifica quando le istruzioni vengono eseguite in base a criteri definiti. È possibile compilare espressioni condizionali che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta. Per vedere tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella IAM User Guide.

> **Note**
>
> Per alcune funzionalità di gestione, Amazon DocumentDB utilizza la tecnologia operativa condivisa con Amazon Relational Database Service (Amazon RDS).
> Per visualizzare un elenco di chiavi di condizione RDS, consulta Chiavi di condizione per Amazon Relational Database Service nel Service Authorization Reference. Per sapere con quali azioni e risorse puoi utilizzare una chiave di condizione, consulta Azioni definite da Amazon Relational Database Service.
> Per visualizzare le chiavi di condizione per i cluster elastici Amazon DocumentDB, consulta Chiavi di condizione per i cluster elastici Amazon DocumentDB nel Service Authorization Reference.

Per visualizzare esempi di policy basate sull'identità di Amazon DocumentDB, consulta. Identity-based esempi di policy per Amazon DocumentDB

ACL in Amazon DocumentDB

Supporta le ACL: no

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni per accedere a una risorsa. Le ACL sono simili alle policy basate su risorse, sebbene non utilizzino il formato del documento di policy JSON.

ABAC con Amazon DocumentDB

> **Note**
>
> ABAC è supportato solo parzialmente per i cluster basati su istanze, ma è supportato per i cluster elastici.

Attribute-based il controllo degli accessi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base ad attributi chiamati tag. È possibile allegare tag alle entità e alle AWS risorse IAM, quindi progettare politiche ABAC per consentire le operazioni quando il tag del principale corrisponde al tag sulla risorsa.

Per controllare l'accesso basato su tag, fornire informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Sì. Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per maggiori informazioni su ABAC, consulta Definizione delle autorizzazioni con autorizzazione ABAC nella Guida per l'utente di IAM. Per visualizzare un tutorial con i passaggi per l'impostazione di ABAC, consulta Utilizzo del controllo degli accessi basato su attributi (ABAC) nella Guida per l'utente di IAM.

Utilizzo di credenziali temporanee con Amazon DocumentDB

Supporta le credenziali temporanee: sì

Le credenziali temporanee forniscono un accesso a breve termine alle AWS risorse e vengono create automaticamente quando utilizzi la federazione o lo scambio di ruoli. AWS consiglia di generare dinamicamente credenziali temporanee anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta Credenziali di sicurezza temporanee in IAM e Servizi AWS compatibili con IAM nella Guida per l'utente IAM.

Cross-service autorizzazioni principali per Amazon DocumentDB

Supporta l'inoltro delle sessioni di accesso (FAS): sì

Le sessioni di accesso diretto (FAS) utilizzano le autorizzazioni del principale che chiama an Servizio AWS, combinate con la richiesta di effettuare richieste Servizio AWS ai servizi a valle. Per i dettagli delle policy relative alle richieste FAS, consulta Forward access sessions.

Ruoli di servizio per Amazon DocumentDB

Supporta i ruoli di servizio: sì

Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Create a role to delegate permissions to an Servizio AWS nella Guida per l'utente IAM.

> **Warning**
>
> La modifica delle autorizzazioni per un ruolo di servizio potrebbe interrompere la funzionalità di Amazon DocumentDB. Modifica i ruoli di servizio solo quando Amazon DocumentDB fornisce indicazioni in tal senso.

Service-linked ruoli per Amazon DocumentDB

> **Note**
>
> Service-linked i ruoli non sono supportati per i cluster basati su istanze ma sono supportati per i cluster elastici.

Un ruolo collegato ai servizi è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo di eseguire un'azione per conto dell'utente. Service-linked i ruoli vengono visualizzati nell'utente Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati al servizio, ma non modificarle.

Per ulteriori informazioni su come creare e gestire i ruoli collegati ai servizi, consulta Servizi AWS supportati da IAM. Trova un servizio nella tabella che includa una colonna **`Yes`** nel Service-linked ruolo. Scegli il collegamento Sì per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Identity-based esempi di policy per Amazon DocumentDB

Per impostazione predefinita, utenti e ruoli non sono autorizzati a creare o modificare risorse Amazon DocumentDB. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy IAM (console) nella Guida per l'utente di IAM.

Per dettagli sulle azioni e sui tipi di risorse definiti da Amazon DocumentDB, incluso il formato degli ARN per ciascuno dei tipi di risorse, consulta Azioni, risorse e chiavi di condizione per Amazon Relational Database Service nel Service Authorization Reference.

> **Note**
>
> Per alcune funzionalità di gestione, Amazon DocumentDB utilizza la tecnologia operativa condivisa con Amazon Relational Database Service (Amazon RDS).
> Per le azioni politiche per i cluster elastici Amazon DocumentDB, consulta Azioni, risorse e chiavi di condizione per Amazon DocumentDB Elastic Clusters nel Service Authorization Reference.

Argomenti

- Best practice per le policy
- Utilizzo della console Amazon DocumentDB
- Consentire agli utenti di visualizzare le loro autorizzazioni

Best practice per le policy

Identity-based le policy determinano se qualcuno può creare, accedere o eliminare le risorse Amazon DocumentDB nel tuo account. Queste operazioni possono comportare costi aggiuntivi per l' Account AWS. Quando si creano o modificano policy basate sull'identità, seguire queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni agli utenti e ai carichi di lavoro, utilizza le policy AWS gestite che concedono le autorizzazioni per molti casi d'uso comuni. Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dal AWS cliente specifiche per i tuoi casi d'uso. Per maggiori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente di IAM.
- Applicazione delle autorizzazioni con privilegio minimo - Quando si impostano le autorizzazioni con le policy IAM, concedere solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegio minimo. Per maggiori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.

- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso - Per limitare l'accesso ad azioni e risorse è possibile aggiungere una condizione alle policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni di servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio CloudFormation. Per maggiori informazioni, consultare la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo dello strumento di analisi degli accessi IAM per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali - Lo strumento di analisi degli accessi IAM convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio (JSON) della policy IAM e alle best practice di IAM. Lo strumento di analisi degli accessi IAM offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per maggiori informazioni, consultare Convalida delle policy per il Sistema di analisi degli accessi IAM nella Guida per l'utente di IAM.
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root Account AWS, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungere le condizioni MFA alle policy. Per maggiori informazioni, consultare Protezione dell'accesso API con MFA nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Utilizzo della console Amazon DocumentDB

Per accedere alla console Amazon DocumentDB (con compatibilità con MongoDB), devi disporre di un set minimo di autorizzazioni. Queste autorizzazioni devono consentirti di elencare e visualizzare i dettagli sulle risorse Amazon DocumentDB presenti nel tuo. Account AWS Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Non è necessario consentire le autorizzazioni minime della console per gli utenti che effettuano chiamate solo all'API AWS CLI o all'API. AWS Al contrario, è opportuno concedere l'accesso solo alle azioni che corrispondono all'operazione API che stanno cercando di eseguire.

Per garantire che utenti e ruoli possano ancora utilizzare la console Amazon DocumentDB, allega anche Amazon DocumentDB `ConsoleAccess` o la policy `ReadOnly` AWS gestita alle entità. Per maggiori informazioni, consulta Aggiunta di autorizzazioni a un utente nella Guida per l'utente di IAM.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono collegate alla relativa identità utente. Questa policy include le autorizzazioni per completare questa azione sulla console o a livello di codice utilizzando l'API or. 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": "*"
        }
    ]
}
```

Risoluzione dei problemi di identità e accesso ad Amazon DocumentDB

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare lavorando con Amazon DocumentDB e IAM.

Argomenti

- Non sono autorizzato a eseguire un'azione in Amazon DocumentDB
- Non sono autorizzato a svolgere attività come: PassRole
- Voglio consentire l'accesso a persone esterne alla mia Account AWS per accedere alle mie risorse Amazon DocumentDB

Non sono autorizzato a eseguire un'azione in Amazon DocumentDB

Se ricevi un errore che indica che non sei autorizzato a eseguire un'operazione, le tue policy devono essere aggiornate per poter eseguire l'operazione.

L'errore di esempio seguente si verifica quando l'utente IAM `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`my-example-widget`* fittizia ma non dispone di autorizzazioni `aws:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 aws:GetWidget on resource: my-example-widget
```

In questo caso, la policy per l'utente `mateojackson` deve essere aggiornata per consentire l'accesso alla risorsa *`my-example-widget`* utilizzando l'azione `aws:`*`GetWidget`*.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Non sono autorizzato a svolgere attività come: PassRole

Se ricevi un messaggio di errore indicante che non sei autorizzato a eseguire l'`iam:PassRole`azione, le tue policy devono essere aggiornate per consentirti di passare un ruolo ad Amazon DocumentDB.

Alcune Servizi AWS consentono di passare un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

Il seguente errore di esempio si verifica quando un utente IAM denominato `marymajor` tenta di utilizzare la console per eseguire un'azione in Amazon DocumentDB. Tuttavia, l'azione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per trasmettere il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire l'accesso a persone esterne alla mia Account AWS per accedere alle mie risorse Amazon DocumentDB

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per servizi che supportano policy basate su risorse o liste di controllo degli accessi (ACL), utilizzare tali policy per concedere alle persone l'accesso alle proprie risorse.

Per maggiori informazioni, consulta gli argomenti seguenti:

- Per scoprire se Amazon DocumentDB supporta queste funzionalità, consulta. Come funziona Amazon DocumentDB con IAM
- Per informazioni su come fornire l'accesso alle risorse di Account AWS cui sei proprietario, consulta Fornire l'accesso a un utente IAM in un altro Account AWS utente di tua proprietà nella IAM User Guide.
- Per sapere come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a risorse Account AWS di proprietà di terzi nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (federazione delle identità) nella Guida per l'utente IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente di IAM.

Gestione delle autorizzazioni di accesso alle tue risorse Amazon DocumentDB

Ogni AWS risorsa è di proprietà di un e Account AWS le autorizzazioni per creare o accedere alle risorse sono regolate da politiche di autorizzazione. Un amministratore di account può allegare politiche di autorizzazione alle identità IAM (ovvero utenti, gruppi e ruoli) e alcuni servizi (come AWS Lambda) supportano anche il collegamento di politiche di autorizzazione alle risorse.

> **Note**
>
> Un amministratore account (o un utente amministratore) è un utente con autorizzazioni di amministratore. Per ulteriori informazioni, consulta Best practice IAM nella Guida per l'utente di IAM.

Argomenti

- Risorse e operazioni di Amazon DocumentDB
- Informazioni sulla proprietà delle risorse
- Gestione dell'accesso alle risorse
- Specificazione degli elementi delle policy: azioni, effetti, risorse e principi
- Specifica delle condizioni in una policy

Risorse e operazioni di Amazon DocumentDB

In Amazon DocumentDB, la risorsa principale è un cluster. Amazon DocumentDB supporta altre risorse che possono essere utilizzate con la risorsa principale, come istanze , gruppi di parametri e sottoscrizioni di eventi. Queste risorse vengono chiamate risorse secondarie.

A risorse e risorse secondarie sono associati nomi Amazon Resource Name (ARN) univoci, come illustrato nella tabella seguente.

Tipo di risorsa	Formato ARN
Cluster	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster:` *`db-cluster-name`*

Tipo di risorsa	Formato ARN
Cluster parameter group (Gruppo di parametri del cluster)	`arn:aws:rds:` *region*`:`*account-id* `:cluster-pg:` *cluster-parameter-group-name*
Snapshot del cluster	`arn:aws:rds:` *region*`:`*account-id* `:cluster-snapshot:` *cluster-snapshot-name*
Istanza	`arn:aws:rds:` *region*`:`*account-id* `:db:`*db-instance-name*
Gruppo di sicurezza	`arn:aws:rds:` *region*`:`*account-id* `:secgrp:`*security-group-name*
Gruppo di sottoreti	`arn:aws:rds:` *region*`:`*account-id* `:subgrp:`*subnet-group-name*

Amazon DocumentDB fornisce una serie di operazioni per lavorare con le risorse di Amazon DocumentDB. Per un elenco di operazioni disponibili, consulta Operazioni.

Informazioni sulla proprietà delle risorse

Il proprietario di una risorsa è colui Account AWS che ha creato una risorsa. Ovvero, il proprietario Account AWS della risorsa è l'entità principale (l'account root, un utente IAM o un ruolo IAM) che autentica la richiesta che crea la risorsa. Negli esempi seguenti viene illustrato il funzionamento:

- Se utilizzi le credenziali del tuo account root Account AWS per creare una risorsa Amazon DocumentDB, ad esempio un'istanza, sei il Account AWS proprietario della risorsa Amazon DocumentDB.
- Se crei un utente IAM nella tua Account AWS e concedi le autorizzazioni per creare risorse Amazon DocumentDB a quell'utente, l'utente può creare risorse Amazon DocumentDB. Tuttavia, il tuo Account AWS, a cui appartiene l'utente, possiede le risorse Amazon DocumentDB.
- Se crei un ruolo IAM nel tuo Account AWS con le autorizzazioni per creare risorse Amazon DocumentDB, chiunque possa assumere il ruolo può creare risorse Amazon DocumentDB. Il tuo Account AWS, a cui appartiene il ruolo, è il proprietario delle risorse Amazon DocumentDB.

Gestione dell'accesso alle risorse

La policy delle autorizzazioni descrive chi ha accesso a cosa. Nella sezione seguente vengono descritte le opzioni disponibili per la creazione di policy relative alle autorizzazioni.

> **Note**
>
> Questa sezione illustra l'utilizzo di IAM nel contesto di Amazon DocumentDB. Non vengono fornite informazioni dettagliate sul servizio IAM. Per la documentazione di IAM completa, consulta Che cos'è IAM? nella Guida per l'utente di IAM. Per informazioni sulla sintassi delle policy IAM e le rispettive descrizioni, consulta Riferimento alle policy IAM di AWS nella Guida per l'utente di IAM.

Le policy collegate a un'identità IAM sono denominate policy basate su identità (policy IAM). Le policy collegate a una risorsa sono denominate policy basate sulle risorse. Amazon DocumentDB supporta solo politiche basate sull'identità (politiche IAM).

Argomenti

- Identity-based politiche (politiche IAM)
- Resource-based politiche

Identity-based politiche (politiche IAM)

Puoi collegare le policy alle identità IAM. Ad esempio, puoi eseguire le operazioni seguenti:

- Allega una politica di autorizzazioni a un utente o a un gruppo nel tuo account: un amministratore dell'account può utilizzare una politica di autorizzazioni associata a un determinato utente per concedere a quell'utente le autorizzazioni per creare una risorsa Amazon DocumentDB, ad esempio un'istanza.
- Collega una policy di autorizzazione a un ruolo (assegnazione di autorizzazioni tra account): per concedere autorizzazioni tra più account, è possibile collegare una policy di autorizzazione basata su identità a un ruolo IAM. Ad esempio, un amministratore può creare un ruolo per concedere autorizzazioni tra account a un altro o a un Account AWS servizio come segue: AWS
  1. L'amministratore dell'account A crea un ruolo IAM e attribuisce una policy di autorizzazione al ruolo che concede le autorizzazioni sulle risorse per l'account A.

2. L'amministratore dell'account A attribuisce una policy di attendibilità al ruolo, identificando l'account B come il principale per tale ruolo.
3. L'amministratore dell'account B può quindi delegare le autorizzazioni per assumere il ruolo a qualsiasi utente nell'account B. In questo modo, gli utenti dell'account B possono creare o accedere alle risorse nell'account A. Il responsabile della politica di fiducia può anche essere un responsabile del AWS servizio se si desidera concedere le autorizzazioni a un AWS servizio per assumere il ruolo.

Per ulteriori informazioni sull'uso di IAM per delegare le autorizzazioni, consulta Access Management nella IAM User Guide (Guida per l'utente di IAM).

Di seguito è riportato un esempio di policy che consente all'utente con l'ID di creare istanze `123456789012` per il tuo. Account AWS La nuova istanza database deve utilizzare un gruppo di opzioni e un gruppo di parametri che inizia con `default` e deve utilizzare il gruppo di sottoreti `default`.

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"
            ]
        }
    ]
}
```

Per ulteriori informazioni sull'utilizzo di policy basate sull'identità con Amazon DocumentDB, consulta. Utilizzo di policy basate sull'identità (politiche IAM) per Amazon DocumentDB Per ulteriori informazioni su utenti, gruppi, ruoli e autorizzazioni, consulta Identità (utenti, gruppi e ruoli) nella Guida per l'utente di IAM.

Resource-based politiche

Anche altri servizi, come Amazon Simple Storage Service (Amazon S3), supportano politiche di autorizzazioni basate sulle risorse. Ad esempio, è possibile associare una policy a un bucket Amazon S3 per gestire le autorizzazioni di accesso a quel bucket. Amazon DocumentDB non supporta policy basate sulle risorse.

Specificazione degli elementi delle policy: azioni, effetti, risorse e principi

Per ogni risorsa Amazon DocumentDB (vediRisorse e operazioni di Amazon DocumentDB), il servizio definisce una serie di operazioni API. Per ulteriori informazioni, consulta Operazioni. Per concedere le autorizzazioni per queste operazioni API, Amazon DocumentDB definisce una serie di azioni che puoi specificare in una policy. L'esecuzione di un'operazione API può richiedere le autorizzazioni per più di un'operazione.

Di seguito sono elencati gli elementi di base di una policy:

- Risorsa: in una policy si utilizza il nome della risorsa Amazon (ARN) per identificare la risorsa a cui si applica la policy stessa.
- Operazione: utilizzi le parole chiave per identificare le operazioni sulla risorsa da permettere o rifiutare. Ad esempio, l'autorizzazione `rds:DescribeDBInstances` concede all'utente le autorizzazioni per eseguire l'operazione `DescribeDBInstances`.
- Effetto: l'effetto prodotto quando l'utente richiede l'operazione specifica, ovvero un'autorizzazione o un rifiuto. USe non concedi esplicitamente (consenti) l'accesso a una risorsa, l'accesso viene implicitamente rifiutato. Puoi anche rifiutare esplicitamente l'accesso a una risorsa per garantire che un utente non possa accedervi, anche se l'accesso viene concesso da un'altra policy.
- Principale - Nelle policy basate su identità (policy IAM), l'utente a cui la policy è collegata è il principale implicito. Per policy basate su risorse, specifichi l'utente, l'account, il servizio o un'altra entità che desideri riceva le autorizzazioni (si applica solo alle policy basate su risorse). Amazon DocumentDB non supporta policy basate sulle risorse.

Per ulteriori informazioni sulla sintassi e le descrizioni delle policy IAM, consulta AWS Riferimento alle policy IAM nella Guida per l'utente di IAM.

Per una tabella che mostra tutte le azioni dell'API Amazon DocumentDB e le risorse a cui si applicano, consulta. Autorizzazioni API Amazon DocumentDB: riferimento ad azioni, risorse e condizioni

Specifica delle condizioni in una policy

Quando si concedono le autorizzazioni, è possibile utilizzare il linguaggio della policy IAM per specificare le condizioni in base a cui la policy deve essere applicata. Ad esempio, potresti decidere che una policy venga applicata solo dopo una data specifica. Per ulteriori informazioni su come specificare le condizioni in un linguaggio di policy, consulta la sezione Condizione nella Guida per l'utente di IAM.

Per esprimere le condizioni è necessario utilizzare chiavi di condizione predefinite. Amazon DocumentDB non dispone di chiavi di contesto specifiche del servizio che possono essere utilizzate in una policy IAM. Per un elenco delle chiavi di contesto di condizione globali disponibili per tutti i servizi, consulta Available global condition keys nella IAM User Guide.

Utilizzo di policy basate sull'identità (politiche IAM) per Amazon DocumentDB

> **Important**
>
> Per alcune funzionalità di gestione, Amazon DocumentDB utilizza una tecnologia operativa condivisa con Amazon RDS. Le chiamate alla console e all'API di Amazon DocumentDB vengono registrate come chiamate effettuate all'API Amazon RDS. AWS CLI
> Per prima cosa, consulta gli argomenti introduttivi che spiegano i concetti e le opzioni di base per la gestione dell'accesso alle tue risorse Amazon DocumentDB. Per ulteriori informazioni, consulta Gestione delle autorizzazioni di accesso alle tue risorse Amazon DocumentDB.

In questo argomento vengono forniti esempi di policy basate su identità in cui un amministratore account può collegare policy di autorizzazione a identità IAM, ovvero utenti, gruppi e ruoli.

Di seguito è riportato un esempio di policy IAM.

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"
            ]
        }
    ]
}
```

La policy include una singola istruzione che specifica le autorizzazioni seguenti per l'utente IAM:

- La policy consente all'utente IAM di creare un'istanza utilizzando l'azione CreateDBInstance (questo vale anche per l'operazione https://docs.aws.amazon.com/cli/latest/reference/rds/create-db-instance.html AWS CLI create-db-instance e per). Console di gestione AWS
- L'elemento `Resource` specifica che l'utente può eseguire azioni in o con altre risorse. Specifica le risorse utilizzando un Amazon Resource Name (ARN). Questo ARN include il nome del servizio a cui appartiene la risorsa (`rds`), il Regione AWS (*indica qualsiasi regione in questo esempio), il numero di account utente (`123456789012` è l'ID utente in questo esempio) e il tipo di risorsa.

  L'elemento `Resource` nell'esempio specifica i seguenti vincoli della policy sulle risorse per l'utente:

  - L'identificatore dell'istanza per la nuova istanza deve iniziare con `test` (per esempio, `testCustomerData1`, `test-region2-data`).
  - Il gruppo di parametri cluster per la nuova istanza database deve iniziare con `default`.
  - Il gruppo di sottoreti per la nuova istanza deve essere il gruppo di sottoreti `default`.

La policy non specifica l'elemento `Principal` poiché in una policy basata su identità l'entità che ottiene l'autorizzazione non viene specificata. Quando si collega una policy a un utente, quest'ultimo è l'entità implicita. Quando colleghi una policy di autorizzazioni a un ruolo IAM, il principale identificato nella policy di attendibilità del ruolo ottiene le autorizzazioni.

Per una tabella che mostra tutte le operazioni dell'API Amazon DocumentDB e le risorse a cui si applicano, consulta. Autorizzazioni API Amazon DocumentDB: riferimento ad azioni, risorse e condizioni

Autorizzazioni necessarie per utilizzare la console Amazon DocumentDB

Affinché un utente possa lavorare con la console Amazon DocumentDB, deve disporre di un set minimo di autorizzazioni. Queste autorizzazioni consentono all'utente di descrivere le proprie risorse Amazon DocumentDB Account AWS e di fornire altre informazioni correlate, tra cui informazioni sulla sicurezza e sulla rete di Amazon EC2.

Se decidi di creare una policy IAM più restrittiva delle autorizzazioni minime richieste, la console non funzionerà come previsto per gli utenti con tale policy IAM. Per garantire che tali utenti possano ancora utilizzare la console Amazon DocumentDB, allega anche la policy `AmazonDocDBConsoleFullAccess` gestita all'utente, come descritto in. AWS politiche gestite per Amazon DocumentDB

Non è necessario consentire le autorizzazioni minime della console per gli utenti che effettuano chiamate solo all'API AWS CLI o all'API Amazon DocumentDB.

Esempi di policy gestite dal cliente

In questa sezione, puoi trovare esempi di policy utente che concedono autorizzazioni per varie azioni di Amazon DocumentDB. Queste politiche funzionano quando si utilizzano azioni API Amazon DocumentDB, AWS SDK o. AWS CLI Se utilizzi la console, sarà necessario concedere autorizzazioni aggiuntive specifiche per quest'ultima, come illustrato in Autorizzazioni necessarie per utilizzare la console Amazon DocumentDB.

Per alcune funzionalità di gestione, Amazon DocumentDB utilizza una tecnologia operativa condivisa con Amazon Relational Database Service (Amazon RDS) e Amazon Neptune.

> **Note**
>
> Tutti gli esempi utilizzano la regione Stati Uniti orientali (Virginia settentrionale) () e contengono ID di account `us-east-1` fittizi.

Esempi

- Esempio 1: consenti a un utente di eseguire qualsiasi azione descrittiva su qualsiasi risorsa Amazon DocumentDB

- Esempio 2: Impedire a un utente di eliminare un'istanza
- Esempio 3: Impedire a un utente di creare un cluster a meno che la crittografia dello storage non sia abilitata

Esempio 1: consenti a un utente di eseguire qualsiasi azione descrittiva su qualsiasi risorsa Amazon DocumentDB

La seguente policy di autorizzazione concede a un utente le autorizzazioni per eseguire tutte le operazioni che iniziano con `Describe`. Queste azioni mostrano informazioni su una risorsa Amazon DocumentDB, ad esempio un'istanza. Il carattere jolly (*) nell'`Resource`elemento indica che le azioni sono consentite per tutte le risorse Amazon DocumentDB di proprietà dell'account.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowRDSDescribe",
         "Effect":"Allow",
         "Action":"rds:Describe*",
         "Resource":"*"
      }
   ]
}
```

Esempio 2: Impedire a un utente di eliminare un'istanza

La seguente policy di autorizzazione assegna le autorizzazioni per impedire a un utente di eliminare un'istanza specifica. Ad esempio, potresti voler negare la possibilità di eliminare le istanze di produzione a qualsiasi utente che non sia un amministratore.

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"
         }
      ]
}
```

Esempio 3: Impedire a un utente di creare un cluster a meno che la crittografia dello storage non sia abilitata

La seguente politica di autorizzazione nega a un utente le autorizzazioni per la creazione di un cluster Amazon DocumentDB a meno che la crittografia dello storage non sia abilitata.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "PreventUnencryptedDocumentDB",
         "Effect": "Deny",
         "Action": "RDS:CreateDBCluster",
         "Condition": {
         "Bool": {
         "rds:StorageEncrypted": "false"
      },
         "StringEquals": {
         "rds:DatabaseEngine": "docdb"
         }
      },
      "Resource": "*"
      }
   ]
}
```

AWS politiche gestite per Amazon DocumentDB

Per aggiungere autorizzazioni a utenti, gruppi e ruoli, è più facile utilizzare policy AWS gestite che scrivere policy da soli. La creazione di policy gestite dai clienti IAM che forniscono al team solo

le autorizzazioni di cui ha bisogno richiede tempo e competenza. Per iniziare rapidamente, puoi utilizzare le nostre politiche AWS gestite. Queste policy coprono i casi d'uso comuni e sono disponibili nell'account Account AWS. Per ulteriori informazioni sulle policy AWS gestite, consulta le policy AWS gestite nella IAM User Guide.

AWS i servizi mantengono e aggiornano le politiche AWS gestite. Non è possibile modificare le autorizzazioni nelle politiche AWS gestite. I servizi occasionalmente aggiungono autorizzazioni aggiuntive a una policy AWS gestita per supportare nuove funzionalità. Questo tipo di aggiornamento interessa tutte le identità (utenti, gruppi e ruoli) a cui è collegata la policy. È molto probabile che i servizi aggiornino una policy AWS gestita quando viene lanciata una nuova funzionalità o quando diventano disponibili nuove operazioni. I servizi non rimuovono le autorizzazioni da una policy AWS gestita, quindi gli aggiornamenti delle policy non comprometteranno le autorizzazioni esistenti.

Inoltre, AWS supporta policy gestite per funzioni lavorative che comprendono più servizi. Ad esempio, la policy `ViewOnlyAccess` AWS gestita fornisce l'accesso in sola lettura a molti AWS servizi e risorse. Quando un servizio lancia una nuova funzionalità, AWS aggiunge autorizzazioni di sola lettura per nuove operazioni e risorse. Per l'elenco e la descrizione delle policy di funzione dei processi, consultare la sezione Policy gestite da AWS per funzioni di processi nella Guida per l'utente di IAM.

Le seguenti policy AWS gestite, che puoi allegare agli utenti del tuo account, sono specifiche di Amazon DocumentDB:

- AmazonDocDBFullAccess— Consente l'accesso completo a tutte le risorse Amazon DocumentDB per la versione root. Account AWS
- AmazonDocDBReadOnlyAccess— Consente l'accesso in sola lettura a tutte le risorse Amazon DocumentDB per la versione root. Account AWS
- AmazonDocDBConsoleFullAccess— Concede l'accesso completo per gestire le risorse dei cluster elastici di Amazon DocumentDB e Amazon DocumentDB utilizzando. Console di gestione AWS
- AmazonDocDBElasticReadOnlyAccess— Consente l'accesso in sola lettura a tutte le risorse del cluster elastico Amazon DocumentDB per la radice. Account AWS
- AmazonDocDBElasticFullAccess— Garantisce l'accesso completo a tutte le risorse del cluster elastico Amazon DocumentDB per la radice. Account AWS

AmazonDocDBFullAccess

Questa policy concede autorizzazioni amministrative che consentono l'accesso completo del principale a tutte le azioni di Amazon DocumentDB. Le autorizzazioni in questa policy sono raggruppate come segue:

- Le autorizzazioni Amazon DocumentDB consentono tutte le azioni di Amazon DocumentDB.
- Alcune delle autorizzazioni di Amazon EC2 incluse in questa policy sono necessarie per convalidare le risorse trasmesse in una richiesta API. Questo serve a garantire che Amazon DocumentDB sia in grado di utilizzare correttamente le risorse con un cluster. Le altre autorizzazioni di Amazon EC2 incluse in questa policy consentono ad Amazon DocumentDB di creare AWS le risorse necessarie per consentire la connessione ai cluster.
- Le autorizzazioni Amazon DocumentDB vengono utilizzate durante le chiamate API per convalidare le risorse passate in una richiesta. Sono necessarie affinché Amazon DocumentDB sia in grado di utilizzare la chiave passata con il cluster Amazon DocumentDB.
- I CloudWatch log sono necessari affinché Amazon DocumentDB sia in grado di garantire che le destinazioni di consegna dei log siano raggiungibili e che siano validi per l'uso dei log da parte dei broker.

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

Questa policy concede autorizzazioni di sola lettura che consentono agli utenti di visualizzare le informazioni in Amazon DocumentDB. I committenti a cui è allegata questa policy non possono effettuare aggiornamenti o eliminare le risorse esistenti, né possono creare nuove risorse Amazon DocumentDB. Ad esempio, i principali con queste autorizzazioni possono visualizzare l'elenco dei cluster e delle configurazioni associati al proprio account, ma non possono modificare la configurazione o le impostazioni di alcun cluster. Le autorizzazioni in questa policy sono raggruppate come segue:

- Le autorizzazioni Amazon DocumentDB consentono di elencare le risorse Amazon DocumentDB, descriverle e ottenere informazioni su di esse.
- Le autorizzazioni Amazon EC2 vengono utilizzate per descrivere Amazon VPC, le sottoreti, i gruppi di sicurezza e gli ENI associati a un cluster.
- Un'autorizzazione Amazon DocumentDB viene utilizzata per descrivere la chiave associata al cluster.

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

Concede l'accesso completo per gestire le risorse di Amazon DocumentDB utilizzando quanto segue Console di gestione AWS :

- Le autorizzazioni Amazon DocumentDB per consentire tutte le azioni del cluster Amazon DocumentDB e Amazon DocumentDB.
- Alcune delle autorizzazioni di Amazon EC2 incluse in questa policy sono necessarie per convalidare le risorse trasmesse in una richiesta API. Questo serve a garantire che Amazon DocumentDB sia in grado di utilizzare correttamente le risorse per il provisioning e la manutenzione del cluster. Le altre autorizzazioni di Amazon EC2 in questa policy consentono ad Amazon DocumentDB di creare AWS le risorse necessarie per consentirti di connetterti ai tuoi cluster come VPCEndpoint.
- AWS KMS le autorizzazioni vengono utilizzate durante le chiamate API per convalidare le risorse passate in una AWS KMS richiesta. Sono necessarie affinché Amazon DocumentDB sia in grado di utilizzare la chiave passata per crittografare e decrittografare i dati inattivi con il cluster elastico Amazon DocumentDB.
- I CloudWatch log sono necessari affinché Amazon DocumentDB sia in grado di garantire che le destinazioni di consegna dei log siano raggiungibili e che siano validi per il controllo e la profilazione dell'uso dei log.

- Le autorizzazioni di Secrets Manager sono necessarie per convalidare il segreto specificato e utilizzarlo per configurare l'utente amministratore per i cluster elastici Amazon DocumentDB.
- Le autorizzazioni Amazon RDS sono necessarie per le azioni di gestione dei cluster Amazon DocumentDB. Per alcune funzionalità di gestione, Amazon DocumentDB utilizza una tecnologia operativa condivisa con Amazon RDS.
- Le autorizzazioni SNS consentono ai titolari di accedere agli abbonamenti e agli argomenti di Amazon Simple Notification Service (Amazon SNS) e di pubblicare messaggi Amazon SNS.
- Le autorizzazioni IAM sono necessarie per creare i ruoli collegati al servizio necessari per la pubblicazione di metriche e log.

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

Questa policy concede autorizzazioni di sola lettura che consentono agli utenti di visualizzare le informazioni elastiche sui cluster in Amazon DocumentDB. I committenti a cui è allegata questa policy non possono effettuare aggiornamenti o eliminare le risorse esistenti, né possono creare nuove risorse Amazon DocumentDB. Ad esempio, i principali con queste autorizzazioni possono visualizzare l'elenco dei cluster e delle configurazioni associati al proprio account, ma non possono modificare la configurazione o le impostazioni di alcun cluster. Le autorizzazioni in questa policy sono raggruppate come segue:

- Le autorizzazioni per i cluster elastici di Amazon DocumentDB consentono di elencare le risorse del cluster elastico di Amazon DocumentDB, descriverle e ottenere informazioni su di esse.

- CloudWatch le autorizzazioni vengono utilizzate per verificare le metriche del servizio.

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

Questa policy concede autorizzazioni amministrative che consentono l'accesso completo del principale a tutte le azioni di Amazon DocumentDB per il cluster elastico Amazon DocumentDB.

Questa policy utilizza i AWS tag (https://docs.aws.amazon.com/tag-editor/latest/userguide/tagging.html) entro le condizioni per definire l'accesso alle risorse. Se stai usando un segreto, deve essere etichettato con una chiave `DocDBElasticFullAccess` e un valore di tag.
Se utilizzi una chiave gestita dal cliente, deve essere contrassegnata con una chiave tag `DocDBElasticFullAccess` e un valore di tag.

Le autorizzazioni in questa policy sono raggruppate come segue:

- Le autorizzazioni elastiche del cluster Amazon DocumentDB consentono tutte le azioni di Amazon DocumentDB.
- Alcune delle autorizzazioni di Amazon EC2 incluse in questa policy sono necessarie per convalidare le risorse trasmesse in una richiesta API. Questo serve a garantire che Amazon DocumentDB sia in grado di utilizzare correttamente le risorse per il provisioning e la manutenzione del cluster. Le altre autorizzazioni di Amazon EC2 in questa policy consentono ad Amazon DocumentDB di creare AWS le risorse necessarie per consentire la connessione ai cluster come un endpoint VPC.
- AWS KMS sono necessarie le autorizzazioni affinché Amazon DocumentDB sia in grado di utilizzare la chiave passata per crittografare e decrittografare i dati inattivi all'interno del cluster elastico Amazon DocumentDB.

  > **Note**
  >
  > La chiave gestita dal cliente deve avere un tag con chiave e un valore di tag. `DocDBElasticFullAccess`

- SecretsManager sono necessarie autorizzazioni per convalidare un determinato segreto e utilizzarlo per configurare l'utente amministratore per i cluster elastici Amazon DocumentDB.

  > **Note**
  >
  > Il segreto utilizzato deve avere un tag con chiave `DocDBElasticFullAccess` e un valore di tag.

- Le autorizzazioni IAM sono necessarie per creare i ruoli collegati al servizio necessari per la pubblicazione di metriche e log.

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

Non puoi collegarti `AmazonDocDBElasticServiceRolePolicy` alle tue AWS Identity and Access Management entità. Questa policy è associata a un ruolo collegato al servizio che consente ad Amazon DocumentDB di eseguire azioni per tuo conto. Per ulteriori informazioni, consulta Service-linked ruoli nei cluster elastici.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/DocDB-Elastic"
                    ]
                }
            }
        }
    ]
}
```

Amazon DocumentDB si aggiorna a AWS policy gestite

Modifica	Descrizione	Data
AmazonDocDBElastic FullAccess, AmazonDoc	Politiche aggiornate per aggiungere azioni di manutenzione in sospeso.	2/11/2025

Modifica	Descrizione	Data
DBConsoleFullAccess - Modifica		
AmazonDocDBElasticFullAccess, AmazonDocDBConsoleFullAccess - Modifica	Politiche aggiornate per aggiungere start/stop e copiare le azioni di snapshot del cluster.	2/21/2024
AmazonDocDBElasticReadOnlyAccess, AmazonDocDBElasticFullAccess - Modifica	Politiche aggiornate per aggiungere `cloudwatch:GetMetricData` azioni.	6/21/2023
AmazonDocDBElasticReadOnlyAccess: nuova policy	Nuova policy gestita per i cluster elastici Amazon DocumentDB.	6/8/2023
AmazonDocDBElasticFullAccess: nuova policy	Nuova policy gestita per i cluster elastici Amazon DocumentDB.	6/5/2023
AmazonDocDB-ElasticServiceRolePolicy: nuova policy	Amazon DocumentDB crea un nuovo ruolo collegato al AWS ServiceRoleForDocDB-Elastic servizio per i cluster elastici Amazon DocumentDB.	11/30/2022
AmazonDocDBConsoleFullAccess- Modifica	Politica aggiornata per aggiungere autorizzazioni globali ed elastiche per i cluster Amazon DocumentDB.	11/30/2022
AmazonDocDBConsoleFullAccess,AmazonDocDBFullAccess, AmazonDocDBReadOnlyAccess - Nuova politica	Avvio del servizio.	1/19/2017

Autorizzazioni API Amazon DocumentDB: riferimento ad azioni, risorse e condizioni

Utilizza le seguenti sezioni come riferimento quando configuri Utilizzo di policy basate sull'identità (politiche IAM) per Amazon DocumentDB e scrivi policy di autorizzazioni da allegare a un'identità IAM (policy basate sull'identità).

Di seguito sono elencate tutte le operazioni dell'API Amazon DocumentDB. Nell'elenco sono incluse le azioni corrispondenti per le quali è possibile concedere le autorizzazioni per eseguire l'azione, la AWS risorsa per la quale è possibile concedere le autorizzazioni e le chiavi di condizione che è possibile includere per un controllo granulare degli accessi. Le operazioni, il valore della risorsa e le condizioni vengono specificati rispettivamente nei campi `Action`, `Resource` e `Condition` della policy. Per ulteriori informazioni sulle condizioni, consulta Specifica delle condizioni in una policy.

Puoi utilizzare chiavi di condizione AWS-wide nelle tue policy di Amazon DocumentDB per esprimere le condizioni. Per un elenco completo delle chiavi AWS-wide, consulta Available Keys nella IAM User Guide.

Puoi testare le politiche IAM con il simulatore di policy IAM. Fornisce automaticamente un elenco di risorse e parametri necessari per ogni AWS azione, incluse le azioni Amazon DocumentDB. Il simulatore di policy IAM determina le autorizzazioni necessarie per ciascuna delle azioni specificate. Per informazioni sul simulatore di policy IAM, consulta Testing IAM Policies with the IAM Policy Simulator nella IAM User Guide.

> **Note**
>
> Per specificare un'operazione, utilizza il prefisso `rds:` seguito dal nome dell'operazione API (ad esempio, `rds:CreateDBInstance`).

Di seguito sono elencate le operazioni dell'API Amazon RDS e le relative azioni, risorse e chiavi di condizione.

Argomenti

- Azioni Amazon DocumentDB che supportano le autorizzazioni a livello di risorsa
- Azioni Amazon DocumentDB che non supportano le autorizzazioni a livello di risorsa

Azioni Amazon DocumentDB che supportano le autorizzazioni a livello di risorsa

Resource-level le autorizzazioni offrono la possibilità di specificare le risorse su cui gli utenti sono autorizzati a eseguire azioni. Amazon DocumentDB offre un supporto parziale per le autorizzazioni a livello di risorsa. Ciò significa che per determinate azioni di Amazon DocumentDB, puoi controllare quando gli utenti possono utilizzare tali azioni in base a condizioni che devono essere soddisfatte o a risorse specifiche che gli utenti possono utilizzare. Ad esempio, puoi concedere agli utenti l'autorizzazione a modificare solo specifiche istanze.

Di seguito sono elencate le operazioni dell'API Amazon DocumentDB e le relative azioni, risorse e chiavi di condizione.

> **Note**
>
> Per alcune funzionalità di gestione, Amazon DocumentDB utilizza una tecnologia operativa condivisa con Amazon RDS. Per ulteriori azioni e autorizzazioni di Amazon DocumentDB, consulta Azioni, risorse e chiavi di condizione per Amazon RDS nel Service Authorization Reference.

Operazioni e azioni dell'API Amazon DocumentDB	Resources	Chiavi di condizione
AddTagsToResource `rds:AddTagsToResource`	Istanza `arn:aws:rds:` *region*`:`*account-id*` :db:`*db-instance-name*	`rds:db-tag`
	Gruppo di sottoreti `arn:aws:rds:` *region*`:`*account-id*` :subgrp:`*subnet-group-name*	`rds:subgrp-tag`
ApplyPendingMaintenanceAction	Istanza `arn:aws:rds:` *region*`:`*account-id*` :db:`*db-instance-name*	`rds:db-tag`

Operazioni e azioni dell'API Amazon DocumentDB	Resources	Chiavi di condizione
`rds:ApplyPendingMaintenanceAction`		
CopyDBClusterSnapshot `rds:CopyDBClusterSnapshot`	Snapshot del cluster `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 parameter group (Gruppo di parametri del cluster) `arn:aws:rds:` *region*`:`*account-id*` :cluster-pg:` *cluster-parameter-group-name*	`rds:cluster-pg-tag`
	Gruppo di sottoreti `arn:aws:rds:` *region*`:`*account-id*` :subgrp:`*subnet-group-name*	`rds:subgrp-tag`

Operazioni e azioni dell'API Amazon DocumentDB	Resources	Chiavi di condizione
CreateDBClusterParameterGroup rds:CreateDBClusterParameterGroup	Cluster parameter group (Gruppo di parametri del cluster) 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
	Snapshot del cluster arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
CreateDBInstance rds:CreateDBInstance	Istanza arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
CreateDBSubnetGroup rds:CreateDBSubnetGroup	Gruppo di sottoreti arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag

Operazioni e azioni dell'API Amazon DocumentDB	Resources	Chiavi di condizione
DeleteDBInstance rds:DeleteDBInstance	Istanza arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
DeleteDBSubnetGroup rds:DeleteDBSubnetGroup	Gruppo di sottoreti arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DescribeDBClusterParameterGroups rds:DescribeDBClusterParameterGroups	Cluster parameter group (Gruppo di parametri del cluster) arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
DescribeDBClusterParameters rds:DescribeDBClusterParameters	Cluster parameter group (Gruppo di parametri del cluster) 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

Operazioni e azioni dell'API Amazon DocumentDB	Resources	Chiavi di condizione
DescribeDBClusterSnapshotAttributes rds:DescribeDBClusterSnapshotAttributes	Snapshot del cluster arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
DescribeDBSubnetGroups rds:DescribeDBSubnetGroups	Gruppo di sottoreti arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DescribePendingMaintenanceActions rds:DescribePendingMaintenanceActions	Istanza arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
FailoverDBCluster rds:FailoverDBCluster	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
ListTagsForResource rds:ListTagsForResource	Istanza arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag

Operazioni e azioni dell'API Amazon DocumentDB	Resources	Chiavi di condizione
	Gruppo di sottoreti 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 parameter group (Gruppo di parametri del cluster) arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
ModifyDBClusterParameterGroup rds:ModifyDBClusterParameterGroup	Cluster parameter group (Gruppo di parametri del cluster) arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
ModifyDBClusterSnapshotAttribute rds:ModifyDBClusterSnapshotAttribute	Snapshot del cluster arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag

Operazioni e azioni dell'API Amazon DocumentDB	Resources	Chiavi di condizione
ModifyDBInstance rds:ModifyDBInstance	Istanza arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
RebootDBInstance rds:RebootDBInstance	Istanza arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
RemoveTagsFromResource rds:RemoveTagsFromResource	Istanza arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
	Gruppo di sottoreti arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
ResetDBClusterParameterGroup rds:ResetDBClusterParameterGroup	Cluster parameter group (Gruppo di parametri del cluster) arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
RestoreDBClusterFromSnapshot rds:RestoreDBClusterFromSnapshot	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag

Operazioni e azioni dell'API Amazon DocumentDB	Resources	Chiavi di condizione
	Snapshot del cluster 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
	Gruppo di sottoreti arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag

Azioni Amazon DocumentDB che non supportano le autorizzazioni a livello di risorsa

Puoi utilizzare tutte le azioni di Amazon DocumentDB in una policy IAM per concedere o negare agli utenti il permesso di utilizzare tale azione. Tuttavia, non tutte le azioni di Amazon DocumentDB supportano autorizzazioni a livello di risorsa, che consentono di specificare le risorse su cui è possibile eseguire un'azione. Le seguenti azioni API Amazon DocumentDB attualmente non supportano le autorizzazioni a livello di risorsa. Pertanto, per utilizzare queste azioni in una policy IAM, devi concedere agli utenti il permesso di utilizzare tutte le risorse per l'azione utilizzando un carattere * jolly per l'elemento nella tua dichiarazione. `Resource`

- `rds:DescribeDBClusterSnapshots`
- `rds:DescribeDBInstances`

Cross-service prevenzione sostitutiva confusa

Il problema confused deputy è un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire un'azione può costringere un'entità maggiormente privilegiata a eseguire l'azione. Nel AWS, l'impersonificazione tra servizi può portare al confuso problema del vice. Cross-service l'imitazione può avvenire quando un servizio (il servizio chiamante) chiama un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso. Per evitare ciò, AWS fornisce strumenti per poterti a proteggere i tuoi dati per tutti i servizi con entità di servizio a cui è stato concesso l'accesso alle risorse del tuo account.

Utilizza `aws:SourceArn` le chiavi di contesto delle condizioni `aws:SourceAccount` globali nelle policy delle risorse per limitare le autorizzazioni che Amazon DocumentDB concede a un altro servizio alla risorsa. Utilizzare `aws:SourceArn` se si desidera consentire l'associazione di una sola risorsa all'accesso tra servizi. Utilizzare `aws:SourceAccount` se si desidera consentire l'associazione di qualsiasi risorsa in tale account all'uso tra servizi.

Il modo più efficace per proteggersi dal problema "confused deputy" è quello di utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa. Se non si conosce l'ARN completo della risorsa o si scelgono più risorse, utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con caratteri jolly (*) per le parti sconosciute dell'ARN. Ad esempio, `arn:aws:`*`servicename`*`:*:`*`123456789012`*`:*`.

Se il valore `aws:SourceArn` non contiene l'ID account, ad esempio un ARN di un bucket Amazon S3, è necessario utilizzare entrambe le chiavi di contesto delle condizioni globali per limitare le autorizzazioni.

Il valore di `aws:SourceArn` deve essere cluster.

L'esempio seguente mostra come è possibile utilizzare le chiavi di contesto della `aws:SourceArn` condizione `aws:SourceAccount` globale in Amazon DocumentDB per evitare il confuso problema sostitutivo.

```
{
  "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"
      }
    }
  }
}
```

Autenticazione tramite identità IAM

Gli utenti e le applicazioni di Amazon DocumentDB possono utilizzare utenti e ruoli IAM per autenticarsi in un cluster Amazon DocumentDB. L'autenticazione IAM di Amazon DocumentDB è un metodo di autenticazione senza password. Inoltre, le applicazioni client non inviano i segreti delle password al cluster Amazon DocumentDB quando utilizzano IAM. roles/users Le connessioni dei client vengono invece autenticate AWS STS utilizzando token di sicurezza temporanei. Non-administrative gli utenti e le applicazioni possono ora utilizzare lo stesso ARN di identità IAM quando si connettono a diversi cluster Amazon DocumentDB e altri servizi. AWS

Puoi anche scegliere di utilizzare l'autenticazione basata su password e IAM per autenticare utenti e applicazioni in un cluster Amazon DocumentDB. L'autenticazione IAM è disponibile solo nella versione 5.0 del cluster basato su istanze Amazon DocumentDB. L'autenticazione IAM tramite gli ARN di identità IAM non è supportata per l'utente primario di Amazon DocumentDB.

> **Note**
>
> L'utente principale può essere autenticato solo utilizzando l'autenticazione basata su password esistente.

Argomenti

- Guida introduttiva all'autenticazione tramite utenti e ruoli IAM

- Configurazione AWS tipi di calcolo per l'autenticazione su Amazon DocumentDB utilizzando AWS IAM
- Monitoraggio delle richieste di autenticazione IAM
- Utilizzo dell'autenticazione IAM
- Driver che supportano IAM
- Domande frequenti sull'autenticazione dell'identità IAM

Guida introduttiva all'autenticazione tramite utenti e ruoli IAM

Gli utenti e i ruoli di Amazon DocumentDB con identità IAM vengono creati e gestiti in un `$external` database.

Creazione di un utente

Connettiti come utente principale, quindi crea un utente e un ruolo IAM:

```
use $external;
db.createUser(
    {
        user: "arn:aws:iam::123456789123:user/iamuser",
        mechanisms: ["MONGODB-AWS"],
        roles: [ { role: "readWrite", db: "readWriteDB" } ]
    }
);
```

In alternativa, aggiungi un utente Amazon DocumentDB utilizzando un ruolo IAM:

```
use $external;
db.createUser(
    {
        user: "arn:aws:iam::123456789123:role/iamrole",
        mechanisms: ["MONGODB-AWS"],
        roles: [ { role: "readWrite", db: "readWriteDB" } ]
    }
);
```

Modifica di un utente o ruolo IAM

Modificare un utente IAM esistente:

```
use $external;
db.updateUser(
  "arn:aws:iam::123456789123:user/iamuser",
  {
    roles: [ { role: "read", db: "readDB" } ]
  }
);
```

Modifica un ruolo IAM esistente:

```
use $external;
db.updateUser(
  "arn:aws:iam::123456789123:role/iamrole",
  {
    roles: [ { role: "read", db: "readDB" } ]
  }
);
```

Per concedere o revocare ruoli a un utente IAM:

```
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" } ]
);
```

Per concedere o revocare ruoli da un ruolo IAM:

```
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" } ]
);
```

Eliminazione di un utente o ruolo IAM

Per eliminare un utente IAM esistente:

```
use $external;
db.dropUser("arn:aws:iam::123456789123:user/iamuser");
```

Per eliminare un ruolo IAM esistente:

```
use $external;
db.dropUser("arn:aws:iam::123456789123:role/iamrole");
```

Configura un URI di connessione per l'autenticazione tramite AWS IAM

Per l'autenticazione tramite AWS IAM, utilizza i seguenti parametri URI: `authSource` as `$external` e `authMechanism` as. `MONGODB-AWS` Se utilizzi un utente IAM, i campi nome utente e password vengono sostituiti rispettivamente da una chiave di accesso e una chiave segreta. Se stai assumendo un ruolo IAM, collegato all'ambiente in cui ti trovi (ad esempio, AWS Lambda funzione, istanza Amazon EC2). Non è necessario passare specificamente alcuna credenziale per l'autenticazione utilizzando il meccanismo. `MONGODB-AWS` Se utilizzi driver MongoDB che supportano il meccanismo di `MONGODB-AWS` autenticazione, i driver hanno anche la capacità di recuperare le credenziali del ruolo IAM dall'istanza di calcolo (ad esempio, Amazon EC2, Lambda function e altri). L'esempio seguente utilizza una shell mongo per l'autenticazione `MONGODB-AWS` tramite il passaggio manuale di una chiave di accesso e una chiave segreta (di un utente IAM) per dimostrare l'autenticazione su Amazon DocumentDB.

L'esempio seguente utilizza il codice Python per l'autenticazione utilizzando `MONGODB-AWS` senza passare esplicitamente alcuna credenziale (utilizzando un ruolo IAM collegato all'ambiente) per dimostrare l'autenticazione con Amazon DocumentDB.

```
##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')
```

L'esempio seguente utilizza una shell mongo per l'autenticazione utilizzando un MONGODB-AWS meccanismo passando manualmente una chiave di accesso e una chiave segreta (di un utente IAM) per dimostrare l'autenticazione con Amazon DocumentDB.

```
$ mongo 'mongodb://<access_key>:<secret_key>@<cluster_endpoint>:<db_port>/test?
authSource=%24external&authMechanism=MONGODB-AWS'
```

L'esempio seguente utilizza una shell mongo per l'autenticazione utilizzando MONGODB-AWS senza passare esplicitamente alcuna credenziale (utilizzando il ruolo IAM collegato all'ambiente) per dimostrare l'autenticazione con Amazon DocumentDB.

```
$ mongo 'mongodb://<cluster_endpoint>:<db_port>/test?authSource=
%24external&authMechanism=MONGODB-AWS'
```

Configurazione AWS tipi di calcolo per l'autenticazione su Amazon DocumentDB utilizzando AWS IAM

Utilizzo di Amazon EC2//AWS LambdaAWS Fargate

Amazon EC2 utilizza le seguenti variabili di ambiente. Se hai un ruolo IAM collegato all'istanza EC2 o un ruolo IAM di esecuzione associato a una funzione Lambda o a un task Amazon ECS, queste variabili vengono compilate automaticamente e il driver può recuperare questi valori dall'ambiente:

```
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
AWS_SESSION_TOKEN
```

Per ulteriori informazioni sulle variabili di ambiente, consulta Utilizzo delle variabili di ambiente Lambda nella Developer Guide. AWS Lambda

Utilizzo di Amazon EKS

L'assegnazione di un ruolo ai tuoi pod Amazon Elastic Kubernetes Service (Amazon EKS) configurerà automaticamente le seguenti due variabili di ambiente:

```
AWS_WEB_IDENTITY_TOKEN_FILE - path of web identity token file
AWS_ROLE_ARN - Name of IAM role to connect with
```

Con l'aiuto di queste variabili, assumi manualmente il ruolo dal tuo codice utilizzando la chiamata SDK per: AWS `AssumeRoleWithWebIdentity`

- Ometti il parametro. `ProviderID`
- Trova il valore del `WebIdentityToken` parametro nel file descritto nella variabile di `AWS_WEB_IDENTITY_TOKEN_FILE` ambiente.

Per ulteriori informazioni su Amazon EKS, consulta What is Amazon EKS nella Amazon EKS User Guide.

Monitoraggio delle richieste di autenticazione IAM

Utilizzo del controllo di Amazon DocumentDB

Vai alla cartella dei log di controllo in Amazon CloudWatch e utilizza diversi modelli di ricerca per ottenere i log per l'autenticazione IAM. Ad esempio, utilizza `{ $.param.mechanism = "MONGODB-AWS" }` come modello di ricerca «Cerca in tutti i flussi di log».

Per ulteriori informazioni sugli eventi supportati nell'auditing, consulta. Verifica degli eventi di Amazon DocumentDB

Utilizzo delle metriche di Amazon CloudWatch

`StsGetCallerIdentityCalls`: questa metrica mostra quante `GetCallerIdentity` chiamate sta effettuando un'istanza Amazon DocumentDB all'endpoint Servizio di token di sicurezza AWS regionalized ().AWS STS Fai riferimento alle specifiche di `MONGODB-AWS` autenticazione per spiegare perché le istanze di database devono effettuare chiamate STS. `GetCallerIdentity`

Utilizzo dell'autenticazione IAM

Se non desideri gestire il nome utente e la password nel tuo database, puoi utilizzare l'autenticazione IAM. L'autenticazione IAM è disponibile solo nella versione 5.0 del cluster basato su istanze Amazon DocumentDB.

L'autenticazione IAM dipende da (). Servizio di token di sicurezza AWS AWS STS Se ricevi un'eccezione di AWS STS limitazione quando usi l'autenticazione IAM, riduci la velocità di connessione.

Per le quote IAM, consulta IAM e le AWS STS quote nella IAM User Guide.

Driver che supportano IAM

I driver che supportano Amazon DocumentDB 5.0 e il meccanismo di `MONGODB-AWS` autenticazione devono funzionare con l'implementazione dell'autenticazione IAM in Amazon DocumentDB.

> **Important**
>
> Esiste una limitazione nota con Node.js i driver precedenti alla versione 6.13.1, che attualmente non sono supportati dall'autenticazione dell'identità IAM per Amazon DocumentDB. Node.js i driver e gli strumenti che utilizzano Node.js driver (ad esempio, mongosh) devono essere aggiornati per utilizzare la versione 6.13.1 o successiva. Node.js

Domande frequenti sull'autenticazione dell'identità IAM

Ci sono esempi a cui posso fare riferimento?

Consulta queste pagine per esempi di casi d'uso e configurazioni:

- In che modo gli utenti umani possono autenticarsi su Amazon DocumentDB utilizzando gli utenti IAM e i ruoli IAM
- Password-less autenticazione ad Amazon DocumentDB utilizzando i ruoli IAM

Ricevo un errore durante l'utilizzo del mio driver Python: «pymongo.errors. ConfigurationError: MONGODB- L'autenticazione richiede pymongo-auth-aws».AWS Come posso risolvere questo problema?

Assicurati di utilizzare la seguente dichiarazione durante l'installazione del driver Python con autenticazione IAM:

```
pip install 'pymongo[aws]'
```

Questo installerà le AWS dipendenze aggiuntive necessarie per il funzionamento dell'autenticazione IAM.

La mia connessione si interromperà quando scadono le credenziali temporanee del mio ruolo IAM?

No, le credenziali IAM temporanee vengono utilizzate solo per stabilire la connessione e l'autenticazione. Quindi tutte le ulteriori autorizzazioni avvengono nel cluster Amazon DocumentDB.

Anche in presenza di credenziali IAM rotate/expire, la connessione non si interromperà né diventerà obsoleta.

Autenticazione tramite SCRAM-SHA-256

Amazon DocumentDB (con compatibilità con MongoDB) supporta il meccanismo di autenticazione Salted Challenge Response Authentication (SCRAM) per l'autenticazione basata su password. Con SCRAM, il server verifica che un client conosca la password di un utente senza trasmetterla al server. Amazon DocumentDB supporta due meccanismi SCRAM: e. `SCRAM-SHA-1` `SCRAM-SHA-256` `SCRAM-SHA-256`utilizza la funzione SHA-256 hash più potente e funziona con la modalità FIPS.

Assegni uno o entrambi i meccanismi SCRAM a un utente quando crei o aggiorni quell'utente. Gli utenti creati con `SCRAM-SHA-1` continuano a funzionare senza modifiche. SCRAM-SHA-256 l'autenticazione è disponibile nelle versioni minori del cluster basate su istanze di Amazon DocumentDB 5.0.1+ e 8.0.1+.

> **Le autenticazioni SCRAM e IAM si escludono a vicenda**
>
> Un utente può avere un meccanismo SCRAM o il `MONGODB-AWS` meccanismo, ma non entrambi. Per l'autenticazione dell'identità IAM, consultaAutenticazione tramite identità IAM.

Argomenti

- Creazione di utenti con meccanismi SCRAM
- Aggiornamento degli utenti esistenti
- Autenticazione con SCRAM-SHA-256

Creazione di utenti con meccanismi SCRAM

Quando si crea un utente, è possibile specificare i meccanismi SCRAM da abilitare con il `mechanisms` campo. Gli esempi seguenti creano utenti con `SCRAM-SHA-1` solo, `SCRAM-SHA-256` solo ed entrambi i meccanismi:

```
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']
});
```

Meccanismi predefiniti

Se non specifichi il `mechanisms` campo, Amazon DocumentDB li abilita entrambi `SCRAM-SHA-1` e `SCRAM-SHA-256` per l'utente. I due comandi seguenti sono equivalenti:

```
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']
});
```

L'abilitazione predefinita di entrambi i meccanismi mantiene la compatibilità con i driver MongoDB precedenti che supportano solo. `SCRAM-SHA-1` Preferisco i driver più recenti e la modalità FIPS. `SCRAM-SHA-256` Se non desideri che nessun utente si autentichi con`SCRAM-SHA-1`, specifica `SCRAM-SHA-256` nel `mechanisms` campo solo per tutti gli utenti.

Aggiornamento degli utenti esistenti

Gli utenti creati con `SCRAM-SHA-1` continuano a essere supportati senza modifiche. Per passare da un utente esistente `SCRAM-SHA-1` a`SCRAM-SHA-256`, aggiorna l'utente con una nuova password e il `SCRAM-SHA-256` meccanismo:

```
db.runCommand({
    updateUser: 'existing',
    pwd: passwordPrompt(),
    mechanisms: ['SCRAM-SHA-256']
});
```

Autenticazione con SCRAM-SHA-256

Il driver si autentica automaticamente utilizzando i parametri della stringa di connessione prima di inviare il primo comando su ogni connessione. Per selezionare `SCRAM-SHA-256` un URI di connessione, imposta il `authMechanism` parametro su: `SCRAM-SHA-256`

```
mongodb://username:password@hostname:port/authenticationDatabase?authMechanism=SCRAM-
SHA-256
```

Puoi anche inviare una richiesta di autenticazione esplicita da mongosh:

```
db.auth({user: 'sha256user', pwd: passwordPrompt(), mechanism: 'SCRAM-SHA-256'})
Enter password:
1
```

Se non specifichi un meccanismo di autenticazione nelle opzioni di connessione, il driver seleziona automaticamente un meccanismo supportato dall'utente del database di destinazione.

Gestione degli utenti di Amazon DocumentDB

In Amazon DocumentDB, gli utenti si autenticano in un cluster con una password. Ogni cluster dispone di credenziali di accesso primarie che vengono stabilite durante la creazione del cluster.

> **Note**
>
> Tutti i nuovi utenti creati prima del 26 marzo 2020 hanno ottenuto i ruoli `dbAdminAnyDatabase`, `readWriteAnyDatabase` e `clusterAdmin`. Ti consigliamo di rivalutare tutti gli utenti e modificare i ruoli in base alle necessità per applicare i privilegi minimi per tutti gli utenti nel cluster.
> Per ulteriori informazioni, consulta Accesso al database tramite Access Control Role-Based.

Utente primario e serviceadmin

Un cluster Amazon DocumentDB appena creato ha due utenti: l'utente principale e l'utente. `serviceadmin`

L'utente principale è un singolo utente privilegiato che può eseguire attività amministrative e creare utenti aggiuntivi con ruoli. Quando ti connetti a un cluster Amazon DocumentDB per la prima volta, devi autenticarti utilizzando le credenziali di accesso primarie. L'utente principale riceve queste autorizzazioni amministrative per un cluster Amazon DocumentDB al momento della creazione del cluster e gli viene concesso il ruolo di. `root`

L' utente `serviceadmin` viene creato implicitamente quando viene creato il cluster. Ogni cluster Amazon DocumentDB ha un `serviceadmin` utente che offre AWS la possibilità di gestire il cluster. Non puoi eseguire l'accesso come `serviceadmin`, né eliminarlo, rinominarlo, modificarne la password o le autorizzazioni. Qualsiasi tentativo comporta la generazione di un errore.

> **Note**
>
> Il primario e `serviceadmin` gli utenti di un cluster Amazon DocumentDB non possono essere eliminati e il ruolo dell'`root`utente primario non può essere revocato.
> Se dimentichi la password dell'utente principale, puoi reimpostarla utilizzando Console di gestione AWS o. AWS CLI

Creazione di utenti aggiuntivi

Dopo esserti connesso come utente principale (o qualsiasi utente con il ruolo`createUser`), puoi creare un nuovo utente, come mostrato di seguito.

```
db.createUser(
    {
        user: "sample-user-1",
        pwd: "password123",
        roles:
            [{"db":"admin", "role":"dbAdminAnyDatabase" }]
    }
)
```

Per visualizzare i dettagli dell'utente, puoi utilizzare il comando `show users` come segue. Puoi inoltre rimuovere gli utenti con il comando `dropUser`. Per ulteriori informazioni, consulta Comandi comuni.

```
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"
     }
    ]
}
```

Nel precedente esempio, il nuovo utente `sample-user-1` viene attribuito al database `admin`. Questo è sempre il caso di un nuovo utente. Amazon DocumentDB non ha il concetto di `authenticationDatabase` e quindi tutta l'autenticazione viene eseguita nel contesto del `admin` database.

Quando si creano utenti, se si omette il db campo quando si specifica il ruolo, Amazon DocumentDB attribuirà implicitamente il ruolo al database in cui viene emessa la connessione. Ad esempio, se la connessione viene emessa in relazione al database `sample-database` ed esegui il comando seguente, l'utente `sample-user-2` verrà creato nel database `admin` e avrà le autorizzazioni `readWrite` per il database `sample-database`.

```
db.createUser(
    {
        user: "sample-user-2",
        pwd: "password123",
        roles:
            ["readWrite"]
    }
)
```

La creazione di utenti con ruoli con ambito in tutti i database (ad esempio, `readInAnyDatabase`) richiede che tu sia nell'ambito del database `admin` durante la creazione dell'utente oppure devi specificare esplicitamente il database per il ruolo durante la creazione dell'utente.

Per cambiare il contesto del database, puoi utilizzare il seguente comando.

```
use admin
```

Per ulteriori informazioni sul controllo degli accessi basato sui ruoli e sull'applicazione dei privilegi minimi tra gli utenti del cluster, consulta Accesso al database tramite Access Control Role-Based.

Password a rotazione automatica per Amazon DocumentDB

Con AWS Secrets Manager, puoi sostituire le credenziali codificate nel tuo codice (comprese le password) con una chiamata API a Secrets Manager per recuperare il segreto a livello di programmazione. Questo approccio aiuta a garantire che il segreto non venga compromesso da qualcuno che esamina il codice, perché semplicemente il segreto non è presente. Inoltre, puoi configurare Secrets Manager affinché ruoti automaticamente il segreto in base a una pianificazione

specificata. In questo modo puoi sostituire i segreti a lungo termine con altri a breve termine, contribuendo a ridurre notevolmente il rischio di compromissione.

Usando Secrets Manager, puoi ruotare automaticamente le tue password di Amazon DocumentDB (ovvero i segreti) utilizzando una funzione fornita da Secrets Manager. AWS Lambda

Per ulteriori informazioni AWS Secrets Manager e sull'integrazione nativa con Amazon DocumentDB, consulta quanto segue:

- Blog: Come ruotare le credenziali di Amazon DocumentDB e Amazon Redshift in Secrets Manager AWS
- Che cos'è Secrets Manager? AWS
- Rotazione dei segreti AWS Secrets Manager
- Credenziali Amazon DocumentDB in Secrets Manager

Accesso al database tramite Access Control Role-Based

Puoi limitare l'accesso alle azioni che gli utenti possono eseguire sui database utilizzando il controllo degli accessi basato sui ruoli (RBAC) in Amazon DocumentDB (con compatibilità con MongoDB). RBAC funziona concedendo uno o più ruoli a un utente. Questi ruoli determinano le operazioni che un utente può eseguire sulle risorse del database. Amazon DocumentDB attualmente supporta sia ruoli integrati con ambito a livello di database, come,,,,`read`,,`readWrite`, `readAnyDatabase``clusterAdmin`, sia ruoli definiti dall'utente che possono essere assegnati ad azioni specifiche e risorse granulari come le raccolte basate sui requisiti.

I casi d'uso più comuni per RBAC includono l'applicazione dei privilegi minimi creando utenti con accesso in sola lettura ai database o alle raccolte in un cluster e progetti di applicazioni multi-tenant che consentono a un singolo utente di accedere a un determinato database o raccolta in un cluster.

> **Note**
>
> Tutti i nuovi utenti creati prima del 26 marzo 2020 hanno ottenuto i ruoli `dbAdminAnyDatabase`, `readWriteAnyDatabase` e `clusterAdmin`. Ti consigliamo di rivalutare tutti gli utenti esistenti e modificare i ruoli in base alle necessità per applicare i privilegi minimi per i cluster.

Argomenti

- Concetti RBAC
- Guida introduttiva ai ruoli integrati in RBAC
- Guida introduttiva ai ruoli definiti dall'utente RBAC
- Connessione ad Amazon DocumentDB come utente
- Comandi comuni
- Differenze funzionali
- Limits
- Accesso al database tramite Role-Based Access Control

Concetti RBAC

Di seguito sono riportati termini e concetti importanti relativi al controllo accessi basato sui ruoli. Per ulteriori informazioni sugli utenti di Amazon DocumentDB, consulta. Gestione degli utenti di Amazon DocumentDB

- Utente: una singola entità in grado di autenticarsi nel database ed eseguire operazioni.
- Password: un segreto utilizzato per autenticare l'utente.
- Ruolo: autorizza un utente a eseguire azioni su uno o più database.
- Database di amministrazione: il database in cui gli utenti sono archiviati e su cui sono autorizzati.
- Database (**db**): lo spazio dei nomi all'interno dei cluster che contiene le raccolte per l'archiviazione dei documenti.

Il comando seguente crea un utente denominato `sample-user`.

```
db.createUser({user: "sample-user", pwd: "abc123", roles: [{role: "read", db: "sample-database"}]})
```

In questo esempio:

- `user: "sample-user"`— Indica il nome utente.
- `pwd: "abc123"`— Indica la password dell'utente.
- `role: "read", "db: "sample-database"`— Indica che l'utente `sample-user` disporrà delle autorizzazioni di lettura in`sample-database`.

```
db.createUser({user: "sample-user", pwd: "abc123", roles: [{role: "read", db: "sample-database"}]})
```

L'esempio seguente mostra l'output dopo aver ottenuto l'utente `sample-user` con `db.getUser(sample-user)`. In questo esempio, l'utente `sample-user` risiede nel database `admin` ma ha il ruolo di lettura per il database `sample-database`.



Quando si creano utenti, se si omette il db campo quando si specifica il ruolo, Amazon DocumentDB attribuirà implicitamente il ruolo al database in cui viene emessa la connessione. Ad esempio, se la connessione viene emessa in relazione al database `sample-database` ed esegui il comando seguente, l'utente `sample-user` verrà creato nel database `admin` e avrà le autorizzazioni `readWrite` per il database `sample-database`.

```
db.createUser({user: "sample-user", pwd: "abc123", roles: ["readWrite"]})
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
   "user":"sample-user",
   "roles":[
      {
         "db":"sample-database",
         "role":"readWrite"
      }
   ]
}
```

La creazione di utenti con ruoli con ambito in tutti i database (ad esempio, `readAnyDatabase`) richiede che tu sia nell'ambito del database `admin` durante la creazione dell'utente oppure devi

specificare esplicitamente il database per il ruolo durante la creazione dell'utente. Per emettere comandi sul database `admin`, puoi utilizzare il comando `use admin`. Per ulteriori informazioni, consulta Comandi comuni.

Guida introduttiva ai ruoli integrati in RBAC

Per aiutarti a iniziare a utilizzare il controllo accessi basato sui ruoli, questa sezione illustra uno scenario di esempio per l'applicazione dei privilegi minimi creando ruoli per tre utenti con funzioni di lavoro diverse.

- `user1` è un nuovo manager che deve essere in grado di visualizzare e accedere a tutti i database in un cluster.
- `user2` è un nuovo dipendente che ha bisogno di accedere a un solo database `sample-database-1`, nello stesso cluster.
- `user3` è un dipendente esistente che deve visualizzare e accedere a un database diverso, `sample-database-2` a cui non aveva accesso prima, nello stesso cluster.

Successivamente, sia `user1` che `user2` lasciano la società e quindi il loro accesso deve essere revocato.

Per creare utenti e concedere ruoli, l'utente per il quale esegui l'autenticazione nel cluster deve disporre di un ruolo associato in grado di eseguire operazioni per `createUser` e `grantRole`. Ad esempio, i ruoli `admin` e `userAdminAnyDatabase` possono entrambi concedere tali capacità, per esempio. Per le operazioni per ruolo, consulta Accesso al database tramite Role-Based Access Control.

> **Note**
>
> In Amazon DocumentDB, tutte le operazioni relative agli utenti e ai ruoli (ad esempio`create`,`get`,,`drop`,`grant`,`revoke`, ecc.) vengono eseguite implicitamente nel `admin` database indipendentemente dal fatto che tu stia emettendo o meno comandi sul database. `admin`

Innanzitutto, per capire quali sono gli utenti e i ruoli correnti nel cluster, puoi eseguire il comando `show users`, come nell'esempio seguente. Vedrai due utenti `serviceadmin` e l'utente principale per il cluster. Questi due utenti sono sempre presenti e non possono essere eliminati. Per ulteriori informazioni, consulta Gestione degli utenti di Amazon DocumentDB.

```
show users
```

Per `user1`, crea un ruolo con accesso in lettura e scrittura a tutti i database dell'intero cluster con il comando seguente.

```
db.createUser({user: "user1", pwd: "abc123", roles: [{role: "readWriteAnyDatabase", db:
 "admin"}]})
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
   "user":"user1",
   "roles":[
      {
         "role":"readWriteAnyDatabase",
         "db":"admin"
      }
   ]
}
```

Per `user2`, crea un ruolo con accesso di sola lettura al database `sample-database-1` con il comando seguente.

```
db.createUser({user: "user2", pwd: "abc123", roles: [{role: "read", db: "sample-
database-1"}]})
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
   "user":"user2",
   "roles":[
      {
         "role":"read",
         "db":"sample-database-1"
      }
   ]
}
```

Per simulare lo scenario nel quale `user3` è un utente esistente, crea innanzitutto l'utente `user3` e quindi assegna un nuovo ruolo a `user3`.

```
db.createUser({user: "user3", pwd: "abc123", roles: [{role: "readWrite", db: "sample-
database-1"}]})
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
   "user":"user3",
   "roles":[
      {
         "role":"readWrite",
         "db":"sample-database-1"
      }
   ]
}
```

Ora che l'utente `user3` è stato creato, assegna a `user3` il ruolo `read` su `sample-database-2`.

```
db.grantRolesToUser("user3", [{role: "read", db: "sample-database-2"}])
```

Infine, `user1` e `user2` lasciano l'azienda ed è necessario che il loro accesso al cluster venga revocato. Puoi farlo rilasciando gli utenti, come segue.

```
db.dropUser("user1")
db.dropUser("user2")
```

Per garantire che tutti gli utenti dispongano dei ruoli appropriati, puoi elencare tutti gli utenti con il comando seguente.

```
show users
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
   "_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"
      }
   ]
}
```

Guida introduttiva ai ruoli definiti dall'utente RBAC

Per aiutarti a iniziare con i ruoli definiti dall'utente, questa sezione illustra uno scenario di esempio in cui viene applicato il privilegio minimo creando ruoli per tre utenti con diverse funzioni lavorative.

In questo esempio, vale quanto segue:

- `user1` è un nuovo manager che deve essere in grado di visualizzare e accedere a tutti i database in un cluster.
- `user2`è un nuovo dipendente che richiede solo l'azione «trova» per un solo database`sample-database-1`, nello stesso cluster.
- `user3`è un dipendente esistente che deve visualizzare e accedere a una raccolta specifica, col2, in un database diverso, a `sample-database-2` cui non aveva accesso prima, nello stesso cluster.

- Per `user1`, crea un ruolo con accesso in lettura e scrittura a tutti i database dell'intero cluster con il comando seguente.

```
db.createUser(
 {
     user: "user1", pwd: "abc123",
     roles: [{role: "readWriteAnyDatabase", db: "admin"}]
 }
 )
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
   "user":"user1",
   "roles":[
      {
         "role":"readWriteAnyDatabase",
         "db":"admin"
      }
   ]
}
```

For`user2`, crea un ruolo con privilegi di 'find' per tutte le raccolte nel database `sample-database-1` con il seguente comando. Tieni presente che questo ruolo garantirebbe che tutti gli utenti associati possano eseguire solo query di ricerca.

```
db.createRole(
{
    role: "findRole",
    privileges: [
    {
        resource: {db: "sample-database-1", collection: ""}, actions: ["find"]
    }],
    roles: []
}
)
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
```

```
   "role":"findRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-1",
            "collection":""
         },
         "actions":[
            "find"
         ]
      }
   ],
   "roles":[

   ]
}
```

Quindi, crea l'utente (`user2`) e assegna all'utente il ruolo creato `findRole` di recente.

```
db.createUser(
{
    user: "user2",
    pwd: "abc123",
    roles: []
})

 db.grantRolesToUser("user2",["findRole"])
```

Per simulare lo `user3` scenario di un utente esistente, create prima l'utente`user3`, quindi create un nuovo ruolo chiamato collectionRole che verrà assegnato nel passaggio successivo. `user3`

Ora puoi assegnare un nuovo ruolo a. `user3` Questo nuovo ruolo consentirà di `user3` inserire, aggiornare, eliminare e trovare l'accesso a una raccolta specifica col2 in`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: []
}
)
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
   "role":"collectionRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-2",
            "collection":"col2"
         },
         "actions":[
            "find",
            "update",
            "insert",
            "remove"
         ]
      }
   ],
   "roles":[

   ]
}
```

Ora che l'utente `user3` è stato creato, puoi concedere `user3` il ruolo`collectionFind`.

```
db.grantRolesToUser("user3",["collectionRole"])
```

Infine, `user1` e `user2` lasciano l'azienda ed è necessario che il loro accesso al cluster venga revocato. Puoi farlo rilasciando gli utenti, come segue.

```
db.dropUser("user1")
```

```
db.dropUser("user2")
```

Per garantire che tutti gli utenti dispongano dei ruoli appropriati, puoi elencare tutti gli utenti con il comando seguente.

```
show users
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
   "_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"
      }
   ]
}
```

Connessione ad Amazon DocumentDB come utente

Quando ti connetti a un cluster Amazon DocumentDB, ti connetti nel contesto di un particolare database. Per impostazione predefinita, se non specifichi un database nella stringa di connessione, ti connetti automaticamente al cluster nell'ambito del database `test`. Tutti i comandi a livello di raccolta come `insert` e `find` sono emessi in relazione alle raccolte nel database `test`.

Per vedere il database in cui ti trovi o, in altre parole, su cui stai emettendo comandi, usa il db comando nella shell mongo, come segue.

Query:

```
db
```

Output:

```
test
```

Sebbene la connessione predefinita possa trovarsi nell'ambito del database `test`, ciò non significa necessariamente che l'utente associato alla connessione sia autorizzato a eseguire operazioni sul database `test`. Nello scenario di esempio precedente, se esegui l'autenticazione come utente `user3`, che ha il ruolo `readWrite` per il database `sample-database-1`, l'ambito predefinito della connessione è il database `test`. Tuttavia, se tenti di inserire un documento in una raccolta nel database`test`, visualizzerai un messaggio di errore di autorizzazione. Questo perché tale utente non è autorizzato a eseguire tale comando su tale database, come illustrato di seguito.

Query:

```
db
```

Output:

```
test
```

Query:

```
db.col.insert({x:1})
```

Output:

```
WriteCommandError({ "ok" : 0, "code" : 13, "errmsg" : "Authorization failure" })
```

Se modifichi l'ambito della connessione al database `sample-database-1`, puoi scrivere nella raccolta per la quale l'utente dispone della relativa autorizzazione.

Query:

```
use sample-database-1
```

Output:

```
switched to db sample-database-1
```

Query:

```
db.col.insert({x:1})
```

Output:

```
WriteResult({ "nInserted" : 1})
```

Quando esegui l'autenticazione in un cluster con un determinato utente, puoi anche specificare il database nella stringa di connessione. In questo modo viene rimossa la necessità di eseguire il comando `use` dopo che l'utente è stato autenticato nel database `admin`.

La seguente stringa di connessione autentica l'utente in relazione al database `admin`, ma l'ambito della connessione sarà relativo al database `sample-database-1`.

```
mongo "mongodb://user3:abc123@sample-cluster.node.us-east-1.docdb.amazonaws.com:27017/
sample-database-2"
```

Comandi comuni

Questa sezione fornisce esempi di comandi comuni che utilizzano il controllo degli accessi basato sui ruoli in Amazon DocumentDB. Devi trovarti nell'ambito del database `admin` per creare e modificare utenti e ruoli. Puoi utilizzare il comando `use admin` per passare al database `admin`.

> **Note**
>
> Modifiche a utenti e ruoli si verificheranno implicitamente nel database `admin`. La creazione di utenti con ruoli con ambito in tutti i database (ad esempio, `readAnyDatabase`) richiede che tu sia nell'ambito del database `admin` (ovvero, `use admin`) durante la creazione dell'utente, oppure devi specificare esplicitamente il database per il ruolo durante la creazione dell'utente (come illustrato nell'esempio 2 in questa sezione).

Esempio 1: creazione di un utente con `read` ruolo per il database. `foo`

```
db.createUser({user: "readInFooBar", pwd: "abc123", roles: [{role: "read", db:
 "foo"}]})
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
   "user":"readInFooBar",
   "roles":[
      {
         "role":"read",
         "db":"foo"
      }
   ]
}
```

Esempio 2: creare un utente con accesso in lettura su tutti i database.

```
db.createUser({user: "readAllDBs", pwd: "abc123", roles: [{role: "readAnyDatabase", db:
 "admin"}]})
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
   "user":"readAllDBs",
   "roles":[
      {
         "role":"readAnyDatabase",
         "db":"admin"
      }
```

```
   ]
}
```

Esempio 3: concedere il `read` ruolo a un utente esistente in un nuovo database.

```
db.grantRolesToUser("readInFooBar", [{role: "read", db: "bar"}])
```

Esempio 4: aggiornamento del ruolo di un utente.

```
db.updateUser("readInFooBar", {roles: [{role: "read", db: "foo"}, {role: "read", db:
 "baz"}]})
```

Esempio 5: revocare l'accesso a un database per un utente.

```
db.revokeRolesFromUser("readInFooBar", [{role: "read", db: "baz"}])
```

Esempio 6: descrivi un ruolo integrato.

```
db.getRole("read", {showPrivileges:true})
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
   "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"
         ]
      }
   }
```

Esempio 7: eliminare un utente dal cluster.

```
db.dropUser("readInFooBar")
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
true
```

Esempio 8: creazione di un ruolo con accesso in lettura e scrittura a una raccolta specifica

```
db.createRole(
{
    role: "collectionRole",
    privileges: [
    {
```

```
        resource: {db: "sample-database-2", collection: "col2"}, actions: ["find",
 "update", "insert", "remove"]
    }],
    roles: []
}
)
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
   "role":"collectionRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-2",
            "collection":"col2"
         },
         "actions":[
            "find",
            "update",
            "insert",
            "remove"
         ]
      }
   ],
   "roles":[

   ]
}
```

Esempio 9: creazione di un utente e assegnazione di un ruolo definito dall'utente

```
db.createUser(
{
    user: "user3",
    pwd: "abc123",
    roles: []
})

db.grantRolesToUser("user3",["collectionRole"])
```

Esempio 10: concedere privilegi aggiuntivi a un ruolo definito dall'utente

```
db.grantPrivilegesToRole(
  "collectionRole",
  [
    {
      resource: { db: "sample-database-1", collection: "col1" },
      actions: ["find", "update", "insert", "remove"]
    }
  ]
)
```

Esempio 11: rimuovere i privilegi da un ruolo definito dall'utente

```
db.revokePrivilegesFromRole(
  "collectionRole",
  [
    {
      resource: { db: "sample-database-1", collection: "col2" },
      actions: ["find", "update", "insert", "remove"]
    }
  ]
)
```

Esempio 12: aggiornamento di un ruolo definito dall'utente esistente

```
db.updateRole(
  "collectionRole",
  {
    privileges: [
    {
        resource: {db: "sample-database-3", collection: "sample-collection-3"},
 actions: ["find", "update", "insert", "remove"]
    }],
    roles: []
  }
)
```

Differenze funzionali

In Amazon DocumentDB, le definizioni di utenti e ruoli sono archiviate nel `admin` database e gli utenti vengono autenticati nel `admin` database. Questa funzionalità differisce da MongoDB Community Edition, ma è coerente con MongoDB Atlas.

Amazon DocumentDB supporta anche i flussi di modifica, che forniscono una sequenza ordinata nel tempo di eventi di modifica che si verificano all'interno delle raccolte del cluster. L'`listChangeStreams`azione viene applicata a livello di cluster (ovvero su tutti i database) e può essere applicata a livello di database e cluster. `modifyChangeStreams`

Limits

La tabella seguente contiene i limiti per il controllo degli Role-Based accessi in Amazon DocumentDB.

Description	Limite
Numero di utenti per cluster	1000
Numero di ruoli associati a un utente	1000
Numero di ruoli definiti dall'utente	100
Numero di risorse associate a un privilegio	100

Accesso al database tramite Role-Based Access Control

Con il controllo accessi basato sui ruoli, puoi creare un utente e concedere uno o più ruoli per determinare le operazioni che l'utente può eseguire in un database o in un cluster.

Di seguito è riportato un elenco di ruoli integrati attualmente supportati in Amazon DocumentDB.

> **Note**
>
> In Amazon DocumentDB 4.0 e 5.0, i `ListDatabase` comandi `ListCollection` and possono opzionalmente utilizzare i `authorizedDatabases` parametri `authorizedCollections` and per elencare le raccolte e i database a cui l'utente è autorizzato ad accedere richiedendo rispettivamente i `listDatabase` ruoli `listCollections` and. Inoltre, gli utenti hanno ora la possibilità di bloccare i propri cursori senza richiedere il ruolo. `KillCursor`

Database user

Nome ruolo	Description	Azioni

`read`	Concede a un utente l'accesso in lettura al database specificato.	`changeStreams`
		`collStats`
		`dbStats`
		`find`
		`killCursors`
		`listIndexes`
		`listCollections`

`readWrite`	Concede all'utente l'accesso in lettura e scrittura al database specificato.	Tutte le operazioni dalle autorizzazioni `read`.
		`createCollection`
		`dropCollection`
		`createIndex`
		`dropIndex`
		`insert`
		`killCursors`
		`listIndexes`
		`listCollections`
		`remove`

Nome ruolo	Description	Azioni
		`update`

Cluster user

Nome ruolo	Description	Azioni

leggi AnyDatabase	Concede a un utente l'accesso in lettura a tutti i database nel cluster.	Tutte le operazioni dalle autorizzazioni `read`.
		`listChangeStreams`
		`listDatabases`

`readWriteAnyDatabase`	Concede a un utente l'accesso in lettura e scrittura a tutti i database nel cluster.	Tutte le operazioni dalle autorizzazioni `readWrite` .
		`listChangeStreams`
		`listDatabases`

`userAdminAnyDatabase`	Concede a un utente la possibilità di assegnare e modificare i ruoli o i privilegi di un utente al database specificato.	`changeCustomData`
		`changePassword`
		`createUser`
		`dropRole`
		`dropUser`
		`grantRole`
		`listDatabases`

Nome ruolo	Description	Azioni

		`revokeRole`
		`viewRole`
		`viewUser`

`dbAdminAnyDatabase`	Concede a un utente la possibilità di eseguire ruoli di amministrazione del database su qualsiasi database specificato.	Tutte le operazioni dalle autorizzazioni `dbAdmin`.
		`dropCollection`
		`listDatabases`
		`listChangeStreams`
		`modifyChangeStreams`

Superuser

Nome ruolo	Description	Azioni
`root`	Concede a un utente l'accesso alle risorse e alle operazioni di tutti i ruoli seguenti combinati: `readWriteAnyDatabase` , `dbAdminAnyDatabase` , `userAdminAnyDatabase` , `clusterAdmin` , `restore` e `backup`.	Tutte le operazioni da `readWriteAnyDatabase` , `dbAdminAnyDatabase` , `userAdminAnyDatabase` , `clusterAdmin` , `restore` e `backup`.

Database administrator

Nome ruolo	Description	Azioni
dbAdmin	Concede a un utente la possibilità di eseguire attività amministrative nel database specificato.	bypassDocumentValidation collMod collStats createCollection createIndex dropCollection dropDatabase dropIndex dbStats find killCursors listIndexes listCollections modifyChangeStreams
dbOwner	Concede a un utente la possibilità di eseguire qualsiasi attività amministrativa nel database specificato combinando i ruoli dbAdmin e readWrite .	Tutte le operazioni da dbAdmin e readWrite .

Cluster administrator

nome del ruolo	Description	Azioni

`clusterAdmin`	Concede a un utente il massimo accesso alla gestione del cluster combinando i ruoli `clusterManager` , `clusterMonitor` e `hostManager` .	Tutte le operazioni da `clusterManager` , `clusterMonitor` e `hostManager` .
		`listChangeStreams`
		`dropDatabase`
		`modifyChangeStreams`

`clusterManager`	Concede a un utente la possibilità di eseguire operazioni di gestione e monitoraggio nel cluster specificato.	`listChangeStreams`
		`listSessions`
		`modifyChangeStreams`
		`replSetGetConfig`

`clusterMonitor`	Concede a un utente la possibilità di avere accesso in sola lettura agli strumenti di monitoraggio.	`collStats`
		`dbStats`
		`find`
		`getParameter`
		`hostInfo`
		`indexStats`

nome del ruolo	Description	Azioni
		killCursors listChangeStreams listCollections listDatabases listIndexes listSessions replSetGetConfig serverStatus top
hostManager	Concede a un utente la possibilità di monitorare e gestire i server.	auditConfigure killCursors killAnyCursor killAnySession killop

Backup administrator

Nome ruolo	Description	Azioni
backup	Concede a un utente l'accesso necessario per eseguire il backup dei dati.	getParameter insert

Nome ruolo	Description	Azioni

		`find`
		`listChangeStreams`
		`listCollections`
		`listDatabases`
		`listIndexes`
		`update`

Nome ruolo	Description	Azioni

`restore`	Concede a un utente l'accesso necessario per ripristinare i dati.	`bypassDocumentValidation`
		`changeCustomData`
		`changePassword`
		`collMod`
		`createCollection`
		`createIndex`
		`createUser`
		`dropCollection`
		`dropRole`
		`dropUser`
		`getParameter`
		`grantRole`
		`find`
		`insert`
		`listCollections`
		`modifyChangeStreams`
		`revokeRole`

Nome ruolo	Description	Azioni

		`remove`
		`viewRole`
		`viewUser`
		`update`

Aggiornamento dei certificati TLS di Amazon DocumentDB

Argomenti

- Aggiornamento dell'applicazione e del cluster Amazon DocumentDB
- Rotazione automatica dei certificati del server
- Risoluzione dei problemi
- Domande frequenti

Il certificato di autorità di certificazione (CA) per i cluster Amazon DocumentDB è stato aggiornato nell'agosto del 2024. Se utilizzi cluster Amazon DocumentDB con Transport Layer Security (TLS) abilitato (impostazione predefinita) e non hai ruotato i certificati dell'applicazione client e del server, sono necessari i seguenti passaggi per mitigare i problemi di connettività tra l'applicazione e i cluster Amazon DocumentDB.

- Fase 1: Scaricare il nuovo certificato CA e aggiornare l'applicazione
- Fase 2: Aggiornare il certificato del server

I certificati CA e server sono stati aggiornati nell'ambito delle migliori pratiche standard di manutenzione e sicurezza per Amazon DocumentDB. Le applicazioni client devono aggiungere i nuovi certificati CA ai propri trust store e le istanze Amazon DocumentDB esistenti devono essere aggiornate per utilizzare i nuovi certificati CA prima di questa data di scadenza.

Aggiornamento dell'applicazione e del cluster Amazon DocumentDB

Attenersi alla procedura descritta in questa sezione per aggiornare il bundle di certificati CA dell'applicazione (Fase 1) e i certificati server del cluster (Fase 2). Prima di applicare le modifiche agli

ambienti di produzione, si consiglia di testare questi passaggi in un ambiente di sviluppo o di gestione temporanea.

> **Note**
>
> È necessario completare i passaggi 1 e 2 Regione AWS in ognuno dei quali sono presenti cluster Amazon DocumentDB.

Fase 1: Scaricare il nuovo certificato CA e aggiornare l'applicazione

Scarica il nuovo certificato CA e aggiorna l'applicazione per utilizzare il nuovo certificato CA per creare connessioni TLS ad Amazon DocumentDB. Scaricare il nuovo bundle di certificati CA da https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem. Questa operazione scarica un file denominato `global-bundle.pem`.

> **Note**
>
> Se accedi al keystore che contiene sia il vecchio certificato CA (`rds-ca-2019-root.pem`) che i nuovi certificati CA (,,`rds-ca-ecc384-g1`) `rds-ca-rsa2048-g1rds-ca-rsa4096-g1`, verifica che il keystore selezioni. `global-bundle`

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

Successivamente, aggiornare le applicazioni per utilizzare il nuovo bundle di certificati. Il nuovo pacchetto CA contiene sia il vecchio certificato CA (rds-ca-2019) che i nuovi certificati CA (rds-ca-rsa2048-g1, rds-ca-rsa4096-g1, rds-ca-ecc384-g1). La presenza di entrambi i certificati CA nel nuovo bundle CA consente di aggiornare l'applicazione e il cluster in due fasi.

Per le applicazioni Java, è necessario creare un nuovo trust store con il nuovo certificato CA. Per istruzioni, vedere la scheda Java nell'Connessione con TLS abilitatoargomento.

Per verificare che l'applicazione utilizzi il bundle di certificati CA più recente, consulta Come posso essere sicuro di utilizzare il bundle CA più recente?. Se già utilizzi il bundle di certificati CA più recente nell'applicazione, puoi passare alla fase 2.

Per esempi di utilizzo di un bundle CA con l'applicazione, consulta Crittografia dei dati in transito e Connessione con TLS abilitato.

> **Note**
>
> Attualmente, MongoDB Go Driver 1.2.1 accetta solo un certificato del server emesso da una CA in `sslcertificateauthorityfile`. Consulta Connessione con TLS abilitato la sezione per la connessione ad Amazon DocumentDB utilizzando Go quando TLS è abilitato.

Fase 2: Aggiornare il certificato del server

Dopo aver aggiornato l'applicazione per utilizzare il nuovo pacchetto CA, il passaggio successivo consiste nell'aggiornare il certificato del server modificando ogni istanza in un cluster Amazon DocumentDB. Per modificare le istanze per utilizzare il nuovo certificato server, vedere le istruzioni riportate di seguito.

Amazon DocumentDB fornisce le seguenti CA per firmare il certificato del server DB per un'istanza DB:

- rds-ca-ecc384-g1 —Utilizza un'autorità di certificazione con algoritmo a chiave privata ECC 384 e algoritmo di firma SHA384. supporta la rotazione automatica dei certificati del server. Questa funzionalità è supportata solo su Amazon DocumentDB 4.0 e 5.0.
- rds-ca-rsa2048-g1: nella maggior parte dei casi utilizza un'autorità di certificazione con algoritmo a chiave privata RSA 2048 e algoritmo di firma SHA256. Regioni AWS supporta la rotazione automatica dei certificati del server.
- rds-ca-rsa4096-g1 —Utilizza un'autorità di certificazione con algoritmo a chiave privata RSA 4096 e algoritmo di firma SHA384. supporta la rotazione automatica dei certificati del server.

> **Note**
>
> Se si utilizza il, è possibile verificare la validità delle autorità di certificazione elencate nella AWS CLI sezione precedente utilizzando describe-certificates. https://docs.aws.amazon.com/cli/latest/reference/docdb/describe-certificates.html

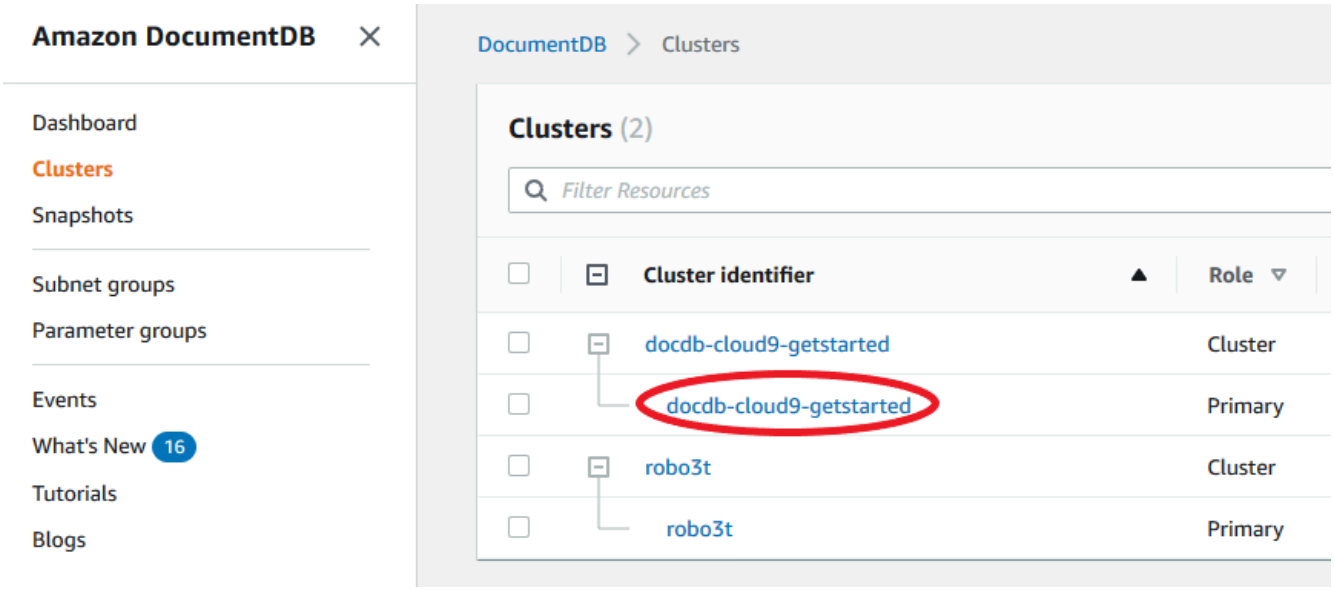
Questi certificati CA sono inclusi nel bundle di certificati regionali e globali. Quando utilizzi la CA rds-ca-rsa2048-g1, rds-ca-rsa4096-g1 o rds-ca-ecc384-g1 con un database, Amazon DocumentDB gestisce il certificato del server DB nel database. Amazon DocumentDB fa ruotare automaticamente il certificato del server DB prima che scada.

> **Note**
>
> Amazon DocumentDB non richiede il riavvio per la rotazione dei certificati se il cluster è in esecuzione con le seguenti versioni di patch del motore:
>
> - Amazon DocumentDB 3.6:1.0.208662 o versioni successive
> - Amazon DocumentDB 4.0:2.0.10179 o versioni successive
> - Amazon DocumentDB 5.0:3.0.4780 o versioni successive
>
> Puoi determinare la versione attuale della patch del motore Amazon DocumentDB eseguendo il seguente comando:. `db.runCommand({getEngineVersion: 1})`
> Prima di aggiornare il certificato del server, assicurati di averlo completatoFase 1.

Using the Console di gestione AWS

Completa i passaggi seguenti per identificare e ruotare il vecchio certificato del server per le tue istanze Amazon DocumentDB esistenti utilizzando. Console di gestione AWS

1. Accedi a e apri la Console di gestione AWS console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nell'elenco delle regioni nell'angolo in alto a destra dello schermo, scegli quella Regione AWS in cui risiedono i tuoi cluster.
3. Nel pannello di navigazione sul lato sinistro della console, scegli Cluster.
4. Potrebbe essere necessario identificare quali istanze sono ancora presenti sul vecchio certificato del server (). `rds-ca-2019` Puoi farlo nella colonna Certificate authority che si trova all'estrema destra della tabella Clusters.
5. Nella tabella Clusters, vedrai la colonna Cluster identifier all'estrema sinistra. Le tue istanze sono elencate in cluster, in modo simile alla schermata seguente.



6. Seleziona la casella a sinistra dell'istanza che ti interessa.
7. Scegliere Actions (Operazioni), quindi Modify (Modifica).
8. In Certificate authority (Autorità di certificazione), selezionare il nuovo certificato del server (ad esempio `rds-ca-rsa2048-g1`) per questa istanza.
9. È possibile visualizzare un riepilogo delle modifiche nella pagina successiva. Considera che è presente un avviso aggiuntivo per ricordare di assicurarsi che l'applicazione stia utilizzando il bundle CA più recente del certificato prima di modificare l'istanza per evitare di causare un'interruzione della connettività.
10. Puoi scegliere di applicare la modifica durante la prossima finestra di manutenzione o applicarla immediatamente. Se si intende modificare immediatamente il certificato del server, utilizzare l'opzione Apply Immediately (Applica immediatamente) .
11. Scegliere Modify instance (Modifica istanza) per completare l'aggiornamento.

Using the AWS CLI

Completa i passaggi seguenti per identificare e ruotare il vecchio certificato del server per le tue istanze Amazon DocumentDB esistenti utilizzando il. AWS CLI

1. Per modificare immediatamente le istanze, eseguire il comando seguente per ogni istanza del cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa2048-g1 --apply-immediately
```

2. Per modificare le istanze del cluster in modo da utilizzare il nuovo certificato CA durante la successiva finestra di manutenzione del cluster, eseguire il comando seguente per ogni istanza del cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa2048-g1 --no-apply-immediately
```

Rotazione automatica dei certificati del server

Amazon DocumentDB supporta la rotazione automatica dei certificati del server. Il certificato del server è il certificato foglia rilasciato a ciascuna istanza del cluster. A differenza dei certificati CA root, i certificati server hanno una validità breve (12 mesi) e Amazon DocumentDB gestisce automaticamente la loro rotazione senza alcuna azione da parte tua. Amazon DocumentDB utilizza la stessa CA root per questa rotazione automatica, quindi non è necessario scaricare un nuovo pacchetto CA.

> **Important**
>
> Quando ti connetti al tuo cluster Amazon DocumentDB, affidati al bundle CA root anziché affidarti direttamente a ciascun certificato del server. Ciò eviterà errori di connessione dopo la rotazione del certificato del server. Consulta Connessione con TLS abilitato.

Amazon DocumentDB tenta di far ruotare il certificato del server nella finestra di manutenzione preferita al termine del periodo di validità del certificato del server. Il nuovo certificato del server è valido per 12 mesi.

Usa il `describe-db-engine-versions` comando e controlla il `SupportsCertificateRotationWithoutRestart` flag per identificare se la versione del motore supporta la rotazione del certificato senza riavvio.

> **Note**
>
> Amazon DocumentDB supporta la rotazione dei certificati del server senza riavvii se il cluster è in esecuzione con le seguenti versioni di patch del motore:
>
> - Amazon DocumentDB 3.6:1.0.208662 o versioni successive
> - Amazon DocumentDB 4.0:2.0.10179 o versioni successive

- Amazon DocumentDB 5.0:3.0.4780 o versioni successive

Puoi determinare la versione attuale della patch del motore Amazon DocumentDB eseguendo questo comando:`db.runCommand({getEngineVersion: 1})`.
Se utilizzi una versione precedente della patch del motore, Amazon DocumentDB ruoterà il certificato del server e pianificherà un evento di riavvio del database nella finestra di manutenzione preferita.

Risoluzione dei problemi

Se riscontri problemi di connessione al cluster nell'ambito della rotazione dei certificati, prova quanto segue:

- Verificare che i client stiano utilizzando il pacchetto di certificati più recente. Consulta Come posso essere sicuro di utilizzare il bundle CA più recente?.
- Verificare che le istanze stiano utilizzando il certificato più recente. Consulta Come faccio a sapere quali delle mie istanze Amazon DocumentDB utilizzano il certificato del old/new server?.
- Verificare che l'applicazione utilizzi il certificato CA più recente. Alcuni driver, come Java e Go, richiedono un codice aggiuntivo per importare più certificati da un bundle di certificati nell'archivio attendibile. Per ulteriori informazioni sulla connessione ad Amazon DocumentDB con TLS, consulta. Connessione programmatica ad Amazon DocumentDB
- Contatta l'assistenza. Se avete domande o problemi, contattateci Supporto.

Domande frequenti

Di seguito sono riportate le risposte ad alcune domande comuni sui certificati TLS.

Cosa succede se ho domande o problemi?

Se avete domande o problemi, contattateci Supporto.

Come faccio a sapere se sto usando TLS per connettermi al mio cluster Amazon DocumentDB?

È possibile stabilire se il cluster utilizza TLS esaminando il parametro `tls` per il gruppo di parametri cluster del cluster. Se il parametro `tls` è impostato su `enabled`, si utilizza il certificato TLS per

connettersi al cluster. Per ulteriori informazioni, consulta Gestione dei gruppi di parametri del cluster Amazon DocumentDB.

Perché si aggiornano i certificati CA e server?

I certificati CA e server di Amazon DocumentDB vengono aggiornati come parte delle migliori pratiche standard di manutenzione e sicurezza per Amazon DocumentDB.

Cosa succede se non intraprendo alcuna azione entro la data di scadenza?

Se utilizzi TLS per connetterti al tuo cluster Amazon DocumentDB con un certificato CA scaduto, le applicazioni che si connettono tramite TLS non saranno più in grado di comunicare con il cluster Amazon DocumentDB.

Amazon DocumentDB non ruoterà automaticamente i certificati del database prima della scadenza. È necessario aggiornare le applicazioni e i cluster per utilizzare i nuovi certificati CA prima o dopo la data di scadenza.

Come faccio a sapere quali delle mie istanze Amazon DocumentDB utilizzano il certificato del old/new server?

Per identificare le istanze di Amazon DocumentDB che utilizzano ancora il vecchio certificato del server, puoi utilizzare Amazon DocumentDB o. Console di gestione AWS AWS CLI

Utilizzo di Console di gestione AWS

Per identificare le istanze nei cluster che utilizzano il certificato precedente

1. Accedi a e apri la Console di gestione AWS console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb

2. Nell'elenco delle regioni nell'angolo in alto a destra dello schermo, scegli la regione Regione AWS in cui risiedono le tue istanze.
3. Nel pannello di navigazione sul lato sinistro della console, scegli Cluster.
4. La colonna Autorità di certificazione (all'estrema destra della tabella) mostra quali istanze sono ancora presenti sul vecchio certificato del server (`rds-ca-2019`) e sul nuovo certificato del server (). `rds-ca-rsa2048-g1`

Utilizzo di AWS CLI

Per identificare le istanze nei cluster che utilizzano il certificato del server precedente, utilizzare il comando `describe-db-clusters` con quanto riportato di seguito.

```
aws docdb describe-db-instances \
    --filters Name=engine,Values=docdb \
    --query 'DBInstances[*].
{CertificateVersion:CACertificateIdentifier,InstanceID:DBInstanceIdentifier}'
```

Come posso modificare le singole istanze nel mio cluster Amazon DocumentDB per aggiornare il certificato del server?

Aggiorna contemporaneamente i certificati del server per tutte le istanze di un determinato cluster. Per modificare le istanze nel cluster, è possibile utilizzare la console o l' AWS CLI.

> **Note**
>
> Prima di aggiornare il certificato del server, assicurati di aver completato la fase 1.

Utilizzo di Console di gestione AWS

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nell'elenco delle regioni nell'angolo in alto a destra dello schermo, scegli quella Regione AWS in cui risiedono i tuoi cluster.
3. Nel pannello di navigazione sul lato sinistro della console, scegli Cluster.
4. La colonna Autorità di certificazione (all'estrema destra della tabella) mostra quali istanze sono ancora presenti sul vecchio certificato del server (). `rds-ca-2019`
5. Nella tabella Clusters, in Identificatore del cluster, seleziona un'istanza da modificare.
6. Scegliere Actions (Operazioni), quindi Modify (Modifica).
7. In Certificate authority (Autorità di certificazione), selezionare il nuovo certificato del server (ad esempio `rds-ca-rsa2048-g1`) per questa istanza.
8. È possibile visualizzare un riepilogo delle modifiche nella pagina successiva. Considera che è presente un avviso aggiuntivo per ricordare di assicurarsi che l'applicazione stia utilizzando

il bundle CA più recente del certificato prima di modificare l'istanza per evitare di causare un'interruzione della connettività.

9. Puoi scegliere di applicare la modifica durante la prossima finestra di manutenzione o applicarla immediatamente.
10. Scegliere Modify instance (Modifica istanza) per completare l'aggiornamento.

Utilizzo di AWS CLI

Completa i passaggi seguenti per identificare e ruotare il vecchio certificato del server per le tue istanze Amazon DocumentDB esistenti utilizzando il. AWS CLI

1. Per modificare immediatamente le istanze, eseguire il comando seguente per ogni istanza del cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa2048-g1 --apply-immediately
```

2. Per modificare le istanze del cluster in modo da utilizzare il nuovo certificato CA durante la successiva finestra di manutenzione del cluster, eseguire il comando seguente per ogni istanza del cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa2048-g1 --no-apply-immediately
```

Cosa succede se si aggiunge una nuova istanza a un cluster esistente?

Tutte le nuove istanze create utilizzano il vecchio certificato server e richiedono connessioni TLS utilizzando il vecchio certificato CA. Tutte le nuove istanze Amazon DocumentDB create dopo il 25 gennaio 2024 utilizzeranno per impostazione predefinita il nuovo certificato rds-ca-rsa2048-g1.

Cosa succede se nel cluster è presente una sostituzione o un failover di istanza?

Se nel cluster è presente una sostituzione di istanza, la nuova istanza creata continua a utilizzare lo stesso certificato server utilizzato in precedenza dall'istanza. Si consiglia di aggiornare contemporaneamente i certificati server per tutte le istanze. Se si verifica un failover nel cluster, viene utilizzato il certificato server sul nuovo primario.

Se non si utilizza TLS per connettersi al cluster, è comunque necessario aggiornare ciascuna delle istanze?

Consigliamo vivamente di abilitare TLS. Nel caso in cui non abiliti TLS, ti consigliamo comunque di ruotare i certificati sulle tue istanze Amazon DocumentDB nel caso in cui prevedi di utilizzare TLS per connetterti ai tuoi cluster in futuro. Se non hai intenzione di utilizzare TLS per connetterti ai tuoi cluster Amazon DocumentDB, non è necessaria alcuna azione.

Se attualmente non uso TLS per connettermi al cluster ma ho intenzione di farlo in futuro, cosa devo fare?

Se hai creato un cluster prima di gennaio 2024, segui i passaggi 1 e 2 della sezione precedente per assicurarti che l'applicazione utilizzi il pacchetto CA aggiornato e che ogni istanza di Amazon DocumentDB utilizzi il certificato server più recente. Se crei un cluster dopo il 25 gennaio 2024, il cluster avrà già il certificato server più recente (rds-ca-rsa2048-g1). Per verificare che l'applicazione utilizzi il bundle di certificati CA più recente, consulta Se non si utilizza TLS per connettersi al cluster, è comunque necessario aggiornare ciascuna delle istanze?.

La scadenza può essere prorogata oltre agosto 2024?

Se le tue applicazioni si connettono tramite TLS, la scadenza non può essere prorogata.

Come posso essere sicuro di utilizzare il bundle CA più recente?

Per verificare di disporre del pacchetto più recente, utilizzate il seguente comando. Per eseguire questo comando, è necessario che java sia installato e che gli strumenti java si trovino nella variabile PATH della shell. Per ulteriori informazioni, vedere Uso di Java

macOS e Amazon Linux

```
keytool -printcert -v -file global-bundle.pem
```

Windows

```
keytool -printcert -v -file global-bundle.p7b
```

Perché viene visualizzato «RDS» nel nome del bundle CA?

Per alcune funzionalità di gestione, come la gestione dei certificati, Amazon DocumentDB utilizza una tecnologia operativa condivisa con Amazon Relational Database Service (Amazon RDS).

Quando scadrà il nuovo certificato?

Il nuovo certificato del server scadrà (generalmente) come segue:

- rds-ca-rsa2048-g1 — Scade nel 2061
- rds-ca-rsa4096-g1 — Scade nel 2121
- rds-ca-ecc384-g1 — Scade nel 2121

Che tipo di errori vedrò se non agisco prima della scadenza del certificato?

I messaggi di errore variano a seconda del conducente. In generale, vedrai errori di convalida del certificato che contengono la stringa «il certificato è scaduto».

Se è stato applicato il nuovo certificato server, è possibile ripristinare il vecchio certificato?

Se devi ripristinare un'istanza al vecchio certificato del server, fallo per tutte le istanze del cluster. È possibile ripristinare il certificato del server per ogni istanza in un cluster utilizzando o il Console di gestione AWS . AWS CLI

Utilizzo di Console di gestione AWS

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nell'elenco delle regioni nell'angolo in alto a destra dello schermo, scegli quella Regione AWS in cui risiedono i tuoi cluster.
3. Nel pannello di navigazione sul lato sinistro della console, scegli Cluster.
4. Nella tabella Cluster, in Identificatore del cluster, seleziona un'istanza da modificare. Scegli Actions (Operazioni), quindi Modify (Modifica).
5. In Certificate authority (Autorità di certificazione), è possibile selezionare il vecchio certificato del server (`rds-ca-2019`).
6. Scegliere Continue (Continua) per visualizzare un riepilogo delle modifiche.
7. In questa pagina, è possibile scegliere di pianificare le modifiche da applicare nella finestra di manutenzione successiva o di applicare le modifiche immediatamente. Effettuare la selezione e scegliere Modify instance (Modifica istanza).

> **Note**
>
> Se si sceglie di applicare le modifiche immediatamente, anche tutte le modifiche incluse nella coda delle modifiche in sospeso saranno applicate. Se nessuna delle modifiche in sospeso richiede tempi di inattività, la scelta di applicarle può provocare tempi di inattività imprevisti.

Utilizzo di AWS CLI

```
aws docdb modify-db-instance --db-instance-identifier <db_instance_name> ca-
certificate-identifier rds-ca-2019 <--apply-immediately | --no-apply-immediately>
```

Se si sceglie `--no-apply-immediately`, la modifica verrà applicata durante la prossima finestra di manutenzione del cluster.

Se ripristino da uno snapshot o un da un momento specifico, sarà disponibile il nuovo certificato del server?

Se ripristini un'istantanea o esegui un ripristino point-in-time dopo agosto 2024, il nuovo cluster creato utilizzerà il nuovo certificato CA.

Cosa succede se ho problemi a connettermi direttamente al mio cluster Amazon DocumentDB da qualsiasi Mac OS?

Mac OS ha aggiornato i requisiti per i certificati affidabili. I certificati affidabili ora devono essere validi per 397 giorni o meno (vedihttps://support.apple.com/en-us/HT211025).

> **Note**
>
> Questa restrizione è stata osservata nelle versioni più recenti di Mac OS.

I certificati di istanza Amazon DocumentDB sono validi per oltre quattro anni, più del massimo consentito per Mac OS. Per connetterti direttamente a un cluster Amazon DocumentDB da un computer con Mac OS, devi consentire certificati non validi durante la creazione della connessione TLS. In questo caso, i certificati non validi indicano che il periodo di validità è superiore a 397

giorni. È necessario comprendere i rischi prima di consentire l'uso di certificati non validi durante la connessione al cluster Amazon DocumentDB.

Per connetterti a un cluster Amazon DocumentDB da Mac OS utilizzando il parametro AWS CLI, usa il parametro. `tlsAllowInvalidCertificates`

```
mongo --tls --host <hostname> --username <username> --password <password> --port 27017
 --tlsAllowInvalidCertificates
```

Convalida della conformità in Amazon DocumentDB

La sicurezza e la conformità di Amazon DocumentDB vengono valutate da revisori di terze parti nell'ambito di diversi programmi di AWS conformità, tra cui:

- System and Organization Controls (SOC) 1, 2 e 3. Per ulteriori informazioni, consulta SOC.
- Programma federale di gestione dei rischi e delle autorizzazioni (FedRAMP). Per ulteriori informazioni, consulta Servizi AWS coperti dal programma di compliance.
- Payment Card Industry Data Security Standard (PCI DSS). Per ulteriori informazioni, consulta PCI DSS.
- ISO 9001, 27001, 27017 e 27018 Per ulteriori informazioni, vedere la certificazione ISO.
- Health Insurance Portability and Accountability Act Business Associate Agreement (HIPAA BAA). Per ulteriori informazioni, consulta Compiance HIPAA.

AWS fornisce un elenco aggiornato di AWS servizi nell'ambito di specifici programmi di conformità nella sezione AWS Services in Scope by Compliance Program.

Third-party i rapporti di audit possono essere scaricati utilizzando AWS Artifact. Per ulteriori informazioni, consulta Download dei report in AWS Artifact.

Per ulteriori informazioni sui programmi di AWS conformità, vedere Programmi di AWS conformità.

La tua responsabilità di conformità quando usi Amazon DocumentDB è determinata dalla sensibilità dei tuoi dati, dagli obiettivi di conformità della tua organizzazione e dalle leggi e dai regolamenti applicabili. Se l'uso di Amazon DocumentDB è soggetto alla conformità a standard come HIPAA o PCI, AWS fornisce risorse per aiutarti a:

- AWS Risorse sulla conformità: una raccolta di cartelle di lavoro e guide che potrebbero essere applicabili al tuo settore e alla tua sede.

- Guide introduttive sulla sicurezza e la conformità: guide all'implementazione che illustrano le considerazioni architetturali e forniscono i passaggi per l'implementazione di ambienti di base incentrati sulla sicurezza e sulla conformità. AWS
- AWS Config: un servizio che valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida del settore e alle normative.
- AWS Security Hub: una visione completa dello stato di sicurezza interno AWS che consente di verificare la conformità agli standard e alle best practice del settore della sicurezza.
- Whitepaper Architecting for HIPAA Security and Compliance: un white paper che descrive come le aziende possono utilizzare per creare applicazioni. AWS HIPAA-compliant

Resilienza in Amazon DocumentDB

L'infrastruttura AWS globale è costruita attorno alle zone Regioni AWS di disponibilità. Regioni AWS forniscono più zone di disponibilità fisicamente separate e isolate, collegate a reti a bassa latenza, alta produttività e altamente ridondanti. Con le zone di disponibilità, è possibile progettare e gestire applicazioni e database che eseguono il failover automatico tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture tradizionali a data center singolo o multiplo.

Un cluster Amazon DocumentDB può essere creato solo in un Amazon VPC con almeno due sottoreti in almeno due zone di disponibilità. Distribuendo le istanze del cluster su almeno due zone di disponibilità, Amazon DocumentDB aiuta a garantire che ci siano istanze disponibili nel cluster nell'improbabile caso di un errore nella zona di disponibilità. Il volume del cluster per il tuo cluster Amazon DocumentDB si estende sempre su tre zone di disponibilità per fornire uno storage durevole con minori possibilità di perdita di dati.

Per ulteriori informazioni sulle zone di disponibilità Regioni AWS e sulle zone di disponibilità, consulta AWS Global Infrastructure.

Oltre all'infrastruttura AWS globale, Amazon DocumentDB offre diverse funzionalità per aiutarti a supportare la resilienza dei dati e le esigenze di backup.

Fault-tolerant e storage con risoluzione automatica

Ogni porzione da 10 GB del volume di archiviazione viene replicata in sei modi, in tre zone di disponibilità. Amazon DocumentDB utilizza uno storage con tolleranza ai guasti che gestisce in modo trasparente la perdita di un massimo di due copie di dati senza influire sulla disponibilità di scrittura del database e fino a tre copie senza influire sulla disponibilità di lettura. Lo storage

di Amazon DocumentDB è inoltre autoriparante; i blocchi di dati e i dischi vengono scansionati continuamente per individuare eventuali errori e sostituiti automaticamente.

Backup e ripristino manuali

Amazon DocumentDB offre la capacità di creare backup completi del cluster per la conservazione e il ripristino a lungo termine. Per ulteriori informazioni, consulta Backup e ripristino in Amazon DocumentDB.

Point-in-time ripristino

Point-in-time il ripristino aiuta a proteggere i cluster Amazon DocumentDB da operazioni di scrittura o eliminazione accidentali. Grazie al ripristino point-in-time, non devi preoccuparti di creare, gestire o programmare i backup on-demand. Per ulteriori informazioni, consulta Ripristino a un determinato punto nel tempo.

Sicurezza dell'infrastruttura in Amazon DocumentDB

In quanto servizio gestito, Amazon DocumentDB è protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi di AWS sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Utilizzi le chiamate API AWS pubblicate per accedere ad Amazon DocumentDB tramite la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di crittografia con Perfect Forward Secrecy (PFS) come DHE (Ephemeral) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Puoi chiamare queste operazioni API da una qualsiasi posizione di rete. Puoi utilizzare le policy di Amazon DocumentDB per controllare l'accesso da specifici endpoint Amazon Virtual Private Cloud (Amazon VPC) o da specifici VPC. In effetti, questo isola l'accesso alla rete a una determinata risorsa Amazon DocumentDB solo dal VPC specifico all'interno della rete. AWS

> **Note**
>
> Amazon DocumentDB non supporta politiche di accesso basate sulle risorse.

API Amazon DocumentDB e interfaccia (endpoint VPC) (AWS PrivateLink)

> **Note**
>
> I cluster elastici Amazon DocumentDB non supportano AWS PrivateLink gli endpoint VPC.

Puoi stabilire una connessione privata tra i tuoi endpoint VPC e l'API Amazon DocumentDB creando un endpoint VPC di interfaccia. Gli endpoint dell'interfaccia sono alimentati da. AWS PrivateLink

Sebbene i cluster basati su istanze Amazon DocumentDB non richiedano una connessione endpoint VPC di interfaccia, AWS PrivateLink consentono di accedere privatamente alle operazioni dell'API di Amazon DocumentDB senza un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione. Direct Connect Le istanze Amazon DocumentDB nel tuo VPC non necessitano di indirizzi IP pubblici per comunicare con gli endpoint API di Amazon DocumentDB per avviare, modificare o terminare istanze di database e cluster di database. Le tue istanze Amazon DocumentDB inoltre non necessitano di indirizzi IP pubblici per utilizzare nessuna delle operazioni API di Amazon DocumentDB disponibili. Il traffico tra il tuo VPC e Amazon DocumentDB non esce dalla rete Amazon.

Ogni endpoint di interfaccia è rappresentato da una o più interfacce di rete elastiche nelle sottoreti. Per ulteriori informazioni, consulta Interfacce di rete elastiche nella Guida per l'utente di Amazon EC2.

Per ulteriori informazioni sugli endpoint VPC, consulta Accedere a un endpoint VPC Servizio AWS usando un'interfaccia nella Guida per l'utente di Amazon Virtual Private Cloud ().AWS PrivateLink Per ulteriori informazioni sulle operazioni di Amazon DocumentDB, consulta Amazon DocumentDB API Reference.

Argomenti

- Considerazioni sugli endpoint VPC dell'
- Disponibilità nelle regioni
- Creazione di un endpoint VPC di interfaccia per l'API Amazon DocumentDB
- Creazione di una policy sugli endpoint VPC per l'API Amazon DocumentDB

Considerazioni sugli endpoint VPC dell'

Prima di configurare un endpoint VPC di interfaccia per gli endpoint API di Amazon DocumentDB, assicurati di esaminare i prerequisiti degli endpoint di interfaccia nella Guida per l'utente di Amazon Virtual Private Cloud ().AWS PrivateLink

Tutte le operazioni dell'API Amazon DocumentDB relative alla gestione delle risorse di Amazon DocumentDB sono disponibili dal tuo VPC utilizzando. AWS PrivateLink

Le policy sugli endpoint VPC sono supportate per gli endpoint API di Amazon DocumentDB. Per impostazione predefinita, l'accesso completo alle operazioni API di Amazon DocumentDB è consentito tramite l'endpoint. Per ulteriori informazioni, consulta Controllare l'accesso agli endpoint VPC utilizzando le policy degli endpoint nella Guida per l'utente di Amazon Virtual Private Cloud ().AWS PrivateLink

Disponibilità nelle regioni

L'API Amazon DocumentDB attualmente supporta gli endpoint VPC nei seguenti settori: Regioni AWS

- Stati Uniti orientali (Ohio)
- Stati Uniti orientali (Virginia settentrionale)
- Stati Uniti occidentali (Oregon)
- Africa (Città del Capo)
- Asia Pacific (Hong Kong)
- Asia Pacifico (Mumbai)
- Asia Pacifico (Hyderabad)
- Asia Pacifico (Osaka)
- Asia Pacifico (Seoul)
- Asia Pacifico (Singapore)
- Asia Pacifico (Sydney)
- Asia Pacifico (Tokyo)
- Canada (Centrale)
- Cina (Pechino)
- Cina (Ningxia)

- Europa (Francoforte)
- Europa (Irlanda)
- Europa (Londra)
- Europa (Parigi)
- Europa (Spagna)
- Europa (Milano)
- Medio Oriente (Emirati Arabi Uniti)
- Sud America (San Paolo)
- AWS GovCloud (US-East)
- AWS GovCloud (US-West)

Creazione di un endpoint VPC di interfaccia per l'API Amazon DocumentDB

Puoi creare un endpoint VPC per l'API Amazon DocumentDB utilizzando la console Amazon VPC o (). AWS Command Line Interface AWS CLI Per ulteriori informazioni, consulta Accedere a un endpoint VPC Servizio AWS tramite interfaccia nella Guida per l'utente di Amazon Virtual Private Cloud ().AWS PrivateLink

Crea un endpoint VPC per l'API Amazon DocumentDB utilizzando il nome del servizio. `com.amazonaws.region.rds`

Ad eccezione Regioni AWS della Cina, se abiliti il DNS privato per l'endpoint, puoi effettuare richieste API ad Amazon DocumentDB con l'endpoint VPC utilizzando il nome DNS predefinito, ad esempio rds.us-east-1.amazonaws.com. Regioni AWS Per la Cina (Pechino) e la Cina (Ningxia) Regioni AWS, puoi effettuare richieste API con l'endpoint VPC utilizzando rispettivamente rds-api.cn-north-1.amazonaws.com .cn e rds-api.cn-northwest-1.amazonaws.com .cn.

Per ulteriori informazioni, consulta Accedere a un endpoint VPC Servizio AWS tramite interfaccia nella Guida per l'utente di Amazon Virtual Private Cloud ().AWS PrivateLink

Creazione di una policy sugli endpoint VPC per l'API Amazon DocumentDB

Puoi allegare una policy sugli endpoint al tuo endpoint VPC che controlli l'accesso all'API Amazon DocumentDB. La policy specifica le informazioni riportate di seguito:

- Il principale che può eseguire operazioni.
- Le azioni che possono essere eseguite.

- Le risorse sui cui si possono eseguire azioni.

Per ulteriori informazioni, consulta Controllare l'accesso agli endpoint VPC utilizzando le policy per gli endpoint nella Guida per l'utente di Amazon Virtual Private Cloud ().AWS PrivateLink

Esempio: policy sugli endpoint VPC per le azioni API di Amazon DocumentDB

Di seguito è riportato un esempio di policy sugli endpoint per l'API Amazon DocumentDB. Se collegata a un endpoint, questa policy consente l'accesso alle azioni API Amazon DocumentDB elencate per tutti i principali su tutte le risorse.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": "*",
      "Effect": "Allow",
      "Action": [
        "rds:CreateDBInstance",
        "rds:ModifyDBInstance",
        "rds:CreateDBSnapshot"
      ],
      "Resource": "*"
    }
  ]
}
```

Esempio: policy sugli endpoint VPC che nega tutti gli accessi a un determinato utente Account AWS

La seguente politica sugli endpoint VPC nega a Account AWS 123456789012 tutti gli accessi alle risorse che utilizzano l'endpoint. La policy consente tutte le azioni compiute da altri. Account AWS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
```

```
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
    }
  ]
}
```

Le migliori pratiche di sicurezza per Amazon DocumentDB

Amazon DocumentDB offre funzionalità di sicurezza tra cui l'isolamento di Amazon VPC, la crittografia a riposo utilizzando AWS KMS, la crittografia in transito tramite TLS e il controllo granulare degli accessi tramite IAM. Utilizza le seguenti best practice per proteggere i cluster, le istanze e i dati di Amazon DocumentDB.

Utilizza i ruoli AWS Identity and Access Management (IAM) per controllare l'accesso alle operazioni API di Amazon DocumentDB. Ciò include le operazioni che creano, modificano o eliminano risorse Amazon DocumentDB come cluster, gruppi di sicurezza e gruppi di parametri. Quando applicazioni o altri AWS servizi devono accedere alle risorse di Amazon DocumentDB, utilizza un ruolo IAM per fornire credenziali temporanee anziché archiviare credenziali a lungo termine nell'applicazione. Quando crei ruoli IAM, applica il principio del privilegio minimo. Per ulteriori informazioni, consulta i ruoli IAM e gli scenari comuni per i ruoli nella IAM User Guide.

- Applica privilegi minimi con il controllo accessi basato sui ruoli.
- Non utilizzare l'utente Account AWS root per gestire le risorse di Amazon DocumentDB. Utilizza invece AWS Identity and Access Management Identity Center per creare utenti umani con credenziali temporanee. Per ulteriori informazioni, consulta gli utenti di IAM Identity Center nella IAM User Guide.
- Concedi a ciascuna identità il set minimo di autorizzazioni necessarie per svolgere le proprie funzioni.
- Usa le policy IAM per gestire le autorizzazioni e segui il principio del privilegio minimo. Per ulteriori informazioni, consulta Best Practice di sicurezza in IAM nella Guida per l'utente di IAM.
- Se utilizzi utenti IAM con credenziali a lungo termine, ruota tali credenziali regolarmente.

> **Warning**
>
> Questo scenario richiede agli utenti IAM un accesso programmatico e credenziali a lungo termine, il che rappresenta un rischio per la sicurezza. Per ridurre questo rischio, si

consiglia di fornire a questi utenti solo le autorizzazioni necessarie per eseguire l'attività e di rimuoverli quando non sono più necessari. Le chiavi di accesso possono essere aggiornate se necessario. Per ulteriori informazioni, consulta la sezione Aggiornamento delle chiavi di accesso nella Guida per l'utente IAM.

- Configura Secrets Manager per ruotare automaticamente i segreti per Amazon DocumentDB. Per ulteriori informazioni, consulta Rotating your secrets e Rotating secrets for Amazon DocumentDB nella Guida per l'utente. AWS Secrets Manager
- Utilizza Transport Layer Security (TLS) e la crittografia dei dati inattivi per crittografare i dati.

Verifica degli eventi di Amazon DocumentDB

Con Amazon DocumentDB (compatibile con MongoDB), puoi controllare gli eventi che sono stati eseguiti nel tuo cluster. Sono esempi di eventi registrati i tentativi di autenticazione riusciti e non riusciti, l'eliminazione di una raccolta in un database o la creazione di un indice. Per impostazione predefinita, il controllo è disabilitato su Amazon DocumentDB e richiede l'attivazione di questa funzionalità.

Quando il controllo è abilitato, Amazon DocumentDB registra gli eventi di Data Definition Language (DDL), Data Manipulation Language (DML), autenticazione, autorizzazione e gestione degli utenti in Amazon Logs. CloudWatch Quando il controllo è abilitato, Amazon DocumentDB esporta i record di controllo del cluster (documenti JSON) in Amazon Logs. CloudWatch Puoi utilizzare Amazon CloudWatch Logs per analizzare, monitorare e archiviare i tuoi eventi di audit di Amazon DocumentDB.

Sebbene Amazon DocumentDB non addebiti costi aggiuntivi per abilitare l'audit, ti vengono addebitate tariffe standard per l'utilizzo dei log. CloudWatch Per informazioni sui prezzi di CloudWatch Logs, consulta i prezzi di Amazon CloudWatch .

La funzionalità di controllo di Amazon DocumentDB è nettamente diversa dall'utilizzo delle risorse di servizio con cui viene monitorato. AWS CloudTrail CloudTrail registra le operazioni eseguite con AWS Command Line Interface (AWS CLI) o Console di gestione AWS su risorse come cluster, istanze, gruppi di parametri e istantanee. Il controllo delle risorse con CloudTrail è attivo per impostazione predefinita e non può essere disabilitato. La funzionalità di controllo di Amazon DocumentDB è una funzionalità opzionale. Registra le operazioni eseguite all'interno del cluster sugli oggetti, ad esempio database, raccolte, indici e utenti.

Argomenti

- Eventi supportati
- Abilitazione del controllo
- Disattivazione del controllo
- Accesso agli eventi di controllo
- Filtraggio degli eventi di controllo DML

Eventi supportati

Il controllo di Amazon DocumentDB supporta le seguenti categorie di eventi:

- Data Definition Language (DDL): include operazioni di gestione del database, connessioni, gestione degli utenti e autorizzazione.
- Eventi di lettura del linguaggio di manipolazione dei dati (letture DML): include `find()` e i vari operatori di aggregazione, operatori aritmetici, operatori booleani e altri operatori di query di lettura.
- Eventi di scrittura del linguaggio di manipolazione dei dati (scritture DML): include operatori e `insert(), update(), delete(), bulkWrite()`

I tipi di evento sono i seguenti:

Tipo di evento	Categoria	Description
`authCheck`	Autorizzazione	Codice risultato 0: successo
		Codice risultato 13: tentativi non autorizzati di eseguire un'operazione.
`authenticate`	Connessione	Tentativi di autentica zione riusciti o non riusciti in una nuova connessione.
`auditConfigure`	DDL	Configurazione del filtro di controllo.

Tipo di evento	Categoria	Description
`createDatabase`	DDL	Creazione di un nuovo database.
`createCollection`	DDL	Creazione di una nuova raccolta all'interno di un database.
`createIndex`	DDL	Creazione di un nuovo indice all'interno di una raccolta.
`dropCollection`	DDL	Eliminazione di una raccolta all'interno di un database.
`dropDatabase`	DDL	Eliminazione di un database.
`dropIndex`	DDL	Eliminazione di un indice all'interno di una raccolta.
`modifyChangeStreams`	DDL	Il flusso di modifiche è stato creato.
`renameCollection`	DDL	Rinominazione di una raccolta all'interno di un database.
`createRole`	Gestione dei ruoli	Creazione di un ruolo.
`dropAllRolesFromDatabase`	Gestione dei ruoli	Eliminazione di tutti i ruoli all'interno di un database.
`dropRole`	Gestione dei ruoli	Abbandono di un ruolo.

Tipo di evento	Categoria	Description
`grantPrivilegesToRole`	Gestione dei ruoli	Concessione di privilegi a un ruolo.
`grantRolesToRole`	Gestione dei ruoli	Assegnazione di ruoli a un ruolo definito dall'utente.
`revokePrivilegesFromRole`	Gestione dei ruoli	Revoca dei privilegi da un ruolo.
`revokeRolesFromRole`	Gestione dei ruoli	Revoca di ruoli da un ruolo definito dall'uten te.
`updateRole`	Gestione dei ruoli	Aggiornamento di un ruolo.
`createUser`	Gestione degli utenti	Creazione di un nuovo utente.
`dropAllUsersFromDatabase`	Gestione degli utenti	Eliminazione di tutti gli utenti all'interno di un database.
`dropUser`	Gestione degli utenti	Eliminazione di un utente esistente.
`grantRolesToUser`	Gestione degli utenti	Assegnazione di ruoli a un utente.
`revokeRolesFromUser`	Gestione degli utenti	Revoca dei ruoli a un utente.
`updateUser`	UserManagement	Aggiornamento di un utente esistente.

Tipo di evento	Categoria	Description
`insert`	Scrittura DML	Inserisce uno o più documenti in una raccolta.
`delete`	Scrittura in formato DML	Elimina uno o più documenti da una raccolta.
`update`	Scrittura in formato DML	Modifica uno o più documenti esistenti in una raccolta.
`bulkWrite`	Scrittura in formato DML	Esegue più operazioni di scrittura con controlli per l'ordine di esecuzione.
`setAuditConfig`	Scrittura in formato DML	Imposta un nuovo filtro per il controllo DML.
`count`	DML letto	Restituisce il numero di documenti che corrisponderebbero a una query find () per la raccolta o la visualizzazione.
`countDocuments`	DML letto	Restituisce il numero di documenti che corrispondono alla query per una raccolta o una visualizzazione.

Tipo di evento	Categoria	Description
`find`	DML letto	Seleziona i documenti in una raccolta o visualizzazione e riporta un cursore ai documenti selezionati.
`getAuditConfig`	DML letto	Recupera il filtro corrente per il controllo DML.
`findAndModify`	Lettura DML e scrittura DML	Modifica e restituisce un singolo documento .
`findOneAndDelete`	Lettura DML e scrittura DML	Elimina un singolo documento in base ai criteri di filtro e ordinamento, restituen do il documento eliminato.
`findOneAndReplace`	Lettura DML e scrittura DML	Sostituisce un singolo documento in base al filtro specificato.
`findOneAndUpdate`	Lettura DML e scrittura DML	Aggiorna un singolo documento in base ai criteri di filtro e ordinamento.
`aggregate`	Lettura DML e scrittura DML	Supporta le API nella pipeline di aggregazi one.

Tipo di evento	Categoria	Description
`distinct`	DML letto	Trova i valori distinti per un campo specificato in una singola raccolta o visualizzazione e restituisce i risultati in un array.

> **Note**
>
> I valori nel campo del parametro del documento relativo agli eventi DML hanno un limite di dimensione di 1 KB. Amazon DocumentDB tronca il valore se supera 1 KB.

> **Note**
>
> Gli eventi di eliminazione TTL non vengono controllati in questo momento.

Abilitazione del controllo

La procedura per abilitare l'audit in un cluster prevede due fasi: Assicurati che entrambi i passaggi siano completati, altrimenti i log di controllo non verranno inviati a CloudWatch Logs.

Passaggio 1. Abilita il parametro del cluster audit_logs

Per abilitare il controllo, è necessario modificare il `audit_logs` parametro nel gruppo di parametri. `audit_logs`è un elenco delimitato da virgole di eventi da registrare. Gli eventi devono essere specificati in minuscolo e non devono esserci spazi vuoti tra gli elementi dell'elenco.

È possibile impostare i seguenti valori per il gruppo di parametri:

Valore	Description	
`ddl`	L'impostazione di questa impostazi	

Valore	Description	
	one consentirà il controllo degli eventi DDL come CreateDatabase, DropDatabase, CreateCollection, DropCollection, CreateIndex, DropIndex, AuthCheck, authenticate, CreateUser, DropUser, grant, revoke, updateUser e drop RolesToUser RolesFromUser AllUsersFromDatabase	
`dml_read`	L'impostazione di questa impostazione consentirà il controllo degli eventi di lettura DML come find, sort count, distinct, group, projecta, unwind, GeoNear, GeoIntersects, GeoWithin e altri operatori di query di lettura MongoDB.	

Valore	Description	
`dml_write`	L'impostazione di questa impostazione consentirà il controllo degli eventi di scrittura DML come insert (), update (), delete () e bulkWrite ()	
`all`	L'impostazione di questa impostazi one consentirà il controllo degli eventi del database, come le query di lettura, le query di scrittura, le azioni del database e le azioni dell'ammi nistratore.	
`none`	L'impostazione di questa impostazione disabiliterà il controllo	

Valore	Description	
`enabled`(versione precedente)	Questa è un'impostazione dei parametri legacy equivalente a 'ddl'. L'impostazione di questa impostazione consentirà il controllo di eventi DDL come CreateDatabase, DropDatabase, CreateCollection, DropCollection, CreateIndex, DropIndex, AuthCheck, authenticate, CreateUser, DropUser, grant, revoke, updateUser e drop. RolesToUser RolesFromUser AllUsersFromDatabase Non è consigliabile utilizzare questa impostazione perché è un'impostazione precedente.	
`disabled` (legacy)	Questa è un'impostazione dei parametri legacy equivalente a 'nessuno'. Non è consigliabile utilizzare questa impostazione perché è un'impostazione precedente.	

> **Note**
>
> Il valore predefinito per il parametro del cluster audit_logs è none (legacy "disabled«).

È inoltre possibile utilizzare i valori sopra indicati in combinazioni.

Valore	Description
`ddl, dml_read`	L'impostazione di questa impostazione consentirà il controllo degli eventi DDL e degli eventi di lettura DML.
`ddl, dml_write`	L'impostazione di questa impostazione consentirà il controllo degli eventi DDL e della scrittura DML
`dml_read, dml_write`	L'impostazione di questa impostazione consentirà il controllo di tutti gli eventi DML

> **Note**
>
> Non è consentito modificare un gruppo di parametri predefinito.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Creazione di gruppi di parametri del cluster Amazon DocumentDB

Dopo la creazione di un gruppo personalizzato di parametri, modificarlo impostando il valore del parametro `audit_logs` su `all`.

- Modifica dei gruppi di parametri del cluster Amazon DocumentDB

Passaggio 2. Abilita l'esportazione di Amazon Logs CloudWatch

Quando il valore del parametro del `audit_logs` cluster è`enabled`,, o `ddl` `dml_read`dml_write, devi anche abilitare Amazon DocumentDB per esportare i log in Amazon CloudWatch. Se ometti uno di questi passaggi, i log di controllo non verranno inviati a. CloudWatch

Quando si crea un cluster, si esegue un ripristino point-in-time o si ripristina un'istantanea, è possibile abilitare i log seguendo questi passaggi. CloudWatch

Using the Console di gestione AWS

Per consentire ad Amazon DocumentDB di esportare i log CloudWatch utilizzando la console, consulta i seguenti argomenti:

- Durante la creazione di un cluster: inCreazione di un cluster e di un'istanza primaria utilizzando Console di gestione AWS, consulta Creazione di un cluster: configurazioni aggiuntive (passaggio 5, esportazioni di log)
- Quando si modifica un cluster esistente: Modifica di un cluster Amazon DocumentDB
- Quando si esegue il ripristino di uno snapshot del cluster: Ripristino da un'istantanea del cluster
- Quando si esegue un ripristino point-in-time: Ripristino a un determinato punto nel tempo

Using the AWS CLI

Per abilitare i log di audit durante la creazione di un nuovo cluster

Il codice seguente crea il cluster `sample-cluster` e abilita CloudWatch i log di controllo.

Example

Per Linux, macOS o 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
```

Per 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
```

Per abilitare i log di audit durante la modifica di un cluster esistente

Il codice seguente modifica il cluster `sample-cluster` e abilita i log di CloudWatch controllo.

Example

Per Linux, macOS o Unix:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

Per Windows:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

L'aspetto dell'output di queste operazioni è simile al seguente (formato JSON).

```
{
    "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"
    }
}
```

Disattivazione del controllo

È possibile disattivare il controllo disattivando l'esportazione dei CloudWatch log e disabilitando il parametro. `audit_logs`

Disattivazione dell'esportazione dei log CloudWatch

È possibile disattivare l'esportazione dei log di controllo utilizzando il Console di gestione AWS o il. AWS CLI

Using the Console di gestione AWS

La seguente procedura utilizza la disattivazione dell' Console di gestione AWS esportazione dei log in Amazon DocumentDB. CloudWatch

Per disabilitare i log di audit

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster. Quindi, scegliere il pulsante a sinistra del nome del cluster per il quale si desidera disabilitare l'esportazione dei log.
3. Scegli Actions (Operazioni), quindi Modify (Modifica).
4. Scorrere verso il basso fino alla sezione Log exports (Esportazioni log) e scegliere Disabled (Disabilitato).
5. Scegli Continua.
6. Esaminare le modifiche, quindi scegliere quando applicare la modifica al cluster.
   - Apply during the next scheduled maintenance window (Applica durante la prossima finestra di manutenzione pianificata)
   - Apply immediately (Applica immediatamente)
7. Scegliere Modify cluster (Modifica cluster).

Using the AWS CLI

Il codice seguente modifica il cluster `sample-cluster` e disabilita CloudWatch i log di controllo.

Example

Per Linux, macOS o Unix:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit"]}'
```

Per Windows:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit"]}'
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "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"
            }
        ]
    }
}
```

Disattivazione del parametro audit_logs

Per disabilitare il parametro `audit_logs` per il cluster, puoi modificarlo in modo che venga utilizzato un gruppo di parametri in cui il parametro `audit_logs` ha il valore `disabled`. In alternativa, puoi modificare il valore del parametro `audit_logs` nel gruppo di parametri del cluster in modo che sia `disabled`.

Per ulteriori informazioni, consulta i seguenti argomenti:

- Modifica di un cluster Amazon DocumentDB
- Modifica dei gruppi di parametri del cluster Amazon DocumentDB

Accesso agli eventi di controllo

Utilizza i seguenti passaggi per accedere ai tuoi eventi di audit su Amazon CloudWatch.

1. Apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Assicurati di trovarti nella stessa regione del cluster Amazon DocumentDB.
3. Nel riquadro di navigazione scegli Logs (Log).
4. Per trovare i log di audit del cluster, dall'elenco individuare e scegliere **`/aws/docdb/`*`yourClusterName`*`/audit`**.

   Gli eventi di audit per ogni istanza sono disponibili nei rispettivi nomi di istanza.

Filtraggio degli eventi di controllo DML

Guida introduttiva al filtro di controllo DML

Gli eventi di controllo DML possono essere filtrati prima di essere scritti su Amazon. CloudWatch Per utilizzare questa funzionalità, è necessario abilitare il registro di controllo e il registro

DML. Amazon DocumentDB supporta il filtraggio `suatype`,, `ecommand`. `user` `namespace` `auditAuthorizationSuccess`

> **Note**
>
> Gli eventi DDL non vengono filtrati.

È possibile abilitare il filtro di controllo in qualsiasi momento specificando il filtro di controllo utilizzando i parametri`setAuditConfig`,`filter`, e `auditAuthorizationSuccess` nell'operazione: `db.adminCommand( { command } )`

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter:
         {
            //filter conditions
         },
      auditAuthorizationSuccess: true | false
   }
)
```

È inoltre possibile recuperare le impostazioni del filtro di controllo eseguendo il comando seguente:

```
db.admin.runCommand( { getAuditConfig: 1})
```

Requisiti di sicurezza

Solo il database users/roles con un'azione privilegiata `auditConfigure` può eseguire i comandi precedenti `admindb` quando si impostano o si elencano i filtri di controllo DML. È possibile utilizzare uno dei ruoli predefiniti di [`clusterAdmin`,`hostManager`,`root`] o creare ruoli personalizzati con `auditConfigure` privilegi. Di seguito è riportato un esempio di utilizzo di ruoli esistenti con il `auditConfigure` privilegio e un esempio con ruoli personalizzati.

Utente con ruolo integrato:

```
use admin
db.createUser(
```

```
  {
    user: "myClusterAdmin",
    pwd: "password123",
    roles: [ { role: "clusterAdmin", db: "admin" } ]
  }
)
```

Utente con ruoli personalizzati:

```
use admin
db.createRole(
   {
     role: "myRole",
     privileges: [
       { resource: { cluster: true }, actions: [ "auditConfigure" ] }
     ],
     roles: []
   }
)
db.createUser(
  {
    user: "myUser",
    pwd: "myPassword",
    roles: [ { role: "myRole", db: "admin" } ]
  }
)
```

Filtraggio dei casi d'uso

Esempio: filtraggio degli eventi in base ai comandi

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
        "$and": [
         {
            "param.command":
               {
                  $in: [ "find","count", "insert", "delete", "update",
 "findandmodify" ]
               }
         }
```

```
         ]
      },
      auditAuthorizationSuccess: true
   }
)
```

Esempio: filtraggio degli eventi in base al nome utente

In questo esempio, verrà registrato solo l'utente «myUser»:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            "param.user":
               {
                  $in: [ "myUser" ]
               }
         }
         ]},
      auditAuthorizationSuccess: true})
```

Esempio: filtraggio per **`atype`**

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {atype: "authCheck"},
      auditAuthorizationSuccess: true
   })
```

> **Note**
>
> Tutti i log DML hanno `authCheck` come. `atype` Solo DDL ne ha una diversa. `atype` Se inserisci un valore diverso da quello `authCheck` di`filter`, non verrà prodotto un accesso DML. CloudWatch

Esempio: filtraggio utilizzando più filtri uniti da operatori

```
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**
>
> Solo al livello superiore `$and` `$or`, e `$nor` sono supportati. Qualsiasi altro operatore non è supportato e causerà un errore.

Esempio: filtraggio per eventi per **`auditAuthorizationSuccess`**

In questo filtro, tutti i comandi che hanno superato con successo l'autorizzazione non verranno registrati:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {},
      auditAuthorizationSuccess: false
   }
)
```

Esempio: filtraggio con e condizioni **`$in` `$nin`**

Quando si utilizza `$in` sia in che`$nin`, il comando non verrà registrato poiché tra le condizioni sarà presente un «e» implicito. In questo esempio, regex bloccherà il `find` comando in modo che non venga registrato nulla:

```
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})
```

Esempio: filtraggio per **`namespace`**

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            "param.ns":
               {
                  $in: [ "test.foo" ]
               }
```

```
        }
        ]},
      auditAuthorizationSuccess: true})
```

Esempio: ripristino del filtro predefinito

Il ripristino del valore predefinito significa che ogni evento di controllo DML verrà registrato. Per ripristinare il filtro al valore predefinito, eseguite il comando seguente:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {},
      auditAuthorizationSuccess: true
   }
)
```

Amazon VPC e Amazon DocumentDB

Amazon Virtual Private Cloud (Amazon VPC) consente di lanciare AWS risorse, come le istanze Amazon DocumentDB, in un cloud privato virtuale (VPC).

Quando utilizzi un VPC, hai il controllo completo sull'ambiente virtuale di rete. Puoi scegliere il tuo intervallo di indirizzi IP, creare sottoreti e configurare liste di routing e di controllo accessi. L'esecuzione del cluster in un VPC non comporta costi aggiuntivi.

Gli account hanno un VPC predefinito. Tutti i nuovi cluster vengono creati nel VPC predefinito, a meno che tu non specifichi diversamente.

Argomenti

- Cluster DocumentDB in un VPC
- Accesso a un cluster Amazon DocumentDB in un VPC
- Crea un IPv4-only VPC da utilizzare con un cluster DocumentDB
- Crea un VPC dual-stack da utilizzare con un cluster DocumentDB

Di seguito, puoi trovare una discussione sulla funzionalità VPC relativa ai cluster Amazon DocumentDB. Per ulteriori informazioni su Amazon VPC, consulta la Guida per l'utente di Amazon VPC.

Cluster DocumentDB in un VPC

Il cluster Amazon DocumentDB si trova in un cloud privato virtuale (VPC). Un VPC è una rete virtuale logicamente isolata da altre reti virtuali in. Cloud AWS Amazon VPC consente di lanciare AWS risorse, come un cluster Amazon DocumentDB o un'istanza Amazon EC2, in un VPC. Il VPC può essere un VPC predefinito fornito con l'account o uno creato da te. Tutti i VPC sono associati al tuo account. AWS

Il VPC predefinito ha tre sottoreti che è possibile usare per isolare le risorse all'interno del VPC. Il VPC predefinito ha anche un gateway Internet che può essere usato per fornire l'accesso alle risorse all'interno del VPC dall'esterno del VPC.

Per gli scenari che coinvolgono cluster Amazon DocumentDB in un VPC e all'esterno di un VPC, consulta. Accesso a un cluster Amazon DocumentDB in un VPC

Argomenti

- Lavorare con i cluster in un VPC
- Lavorare con i gruppi di sottorete
- Indirizzamento IP di Amazon DocumentDB
- Creazione di un cluster in un VPC

Lavorare con i cluster in un VPC

Ecco alcuni suggerimenti su come lavorare con un cluster in un VPC:

- Il VPC deve avere almeno due sottoreti. Queste sottoreti devono trovarsi in due diverse zone di disponibilità nel luogo in Regione AWS cui si desidera distribuire il cluster. Una sottorete è un segmento dell'intervallo di indirizzi IP di un VPC che puoi specificare e che puoi utilizzare per raggruppare i cluster in base alle tue esigenze operative e di sicurezza.
- Se desideri che il tuo cluster nel VPC sia accessibile al pubblico, assicurati di attivare gli attributi VPC e. `DNS hostnames` `DNS resolution`
- Il tuo VPC deve avere un gruppo di sottorete creato da te. Crei un gruppo di sottoreti specificando le sottoreti che hai creato. Amazon DocumentDB sceglie una sottorete e un indirizzo IP all'interno di quel gruppo di sottorete da associare all'istanza principale del cluster. L'istanza principale utilizza la zona di disponibilità che contiene la sottorete.
- Il tuo VPC deve disporre di un gruppo di sicurezza VPC che consenta l'accesso al cluster.

Per ulteriori informazioni, consulta Accesso a un cluster Amazon DocumentDB in un VPC.

- I blocchi CIDR in ciascuna delle tue sottoreti devono essere sufficientemente grandi da contenere indirizzi IP di riserva che Amazon DocumentDB possa utilizzare durante le attività di manutenzione, inclusi il failover e la scalabilità del calcolo. Ad esempio, un intervallo come 10.0.0. 0/24 e 10.0.1. 0/24 è in genere abbastanza grande.
- Un VPC può avere un `instance tenancy` attributo pari a `default` o`dedicated`. Tutti i VPC predefiniti hanno l'`instance tenancy`attributo impostato su default e un VPC predefinito può supportare qualsiasi classe di istanza.

  Se scegli di avere il cluster in un VPC dedicato in cui l'`instance tenancy`attributo è impostato su`dedicated`, la classe di istanza del cluster deve essere uno dei tipi di istanza dedicati di Amazon EC2 approvati. Ad esempio, l'istanza dedicata `r5.large` EC2 corrisponde alla classe dell'`db.r5.large`istanza. Per informazioni sulla tenancy delle istanze in un VPC, consulta le istanze Amazon EC2 nella Amazon Elastic Compute Cloud User Guide.

  Per ulteriori informazioni sui tipi di istanze che possono essere in un'istanza dedicata, consulta Istanze dedicate Amazon EC2 nella pagina dei prezzi Amazon EC2.

  > **Note**
  >
  > Quando si imposta l'`instance tenancy`attributo su `dedicated` per un cluster, non si garantisce che il cluster venga eseguito su un host dedicato.

Lavorare con i gruppi di sottorete

Le sottoreti sono segmenti di un intervallo di indirizzi IP di un VPC che si designa per raggruppare le risorse in base alle esigenze operative e di sicurezza. Un gruppo di sottoreti è una raccolta di sottoreti (in genere private) che crei in un VPC e che poi definisci per i tuoi cluster. Utilizzando un gruppo di sottoreti, puoi specificare un particolare VPC durante la creazione di cluster utilizzando l'API o Amazon DocumentDB. AWS CLI Se utilizzi la console, puoi scegliere il VPC e i gruppi di sottorete che desideri usare.

Ogni gruppo di sottoreti deve avere sottoreti in almeno due zone di disponibilità in una determinata zona. Regione AWS Quando si crea un cluster in un VPC, si sceglie un gruppo di sottorete per esso. Dal gruppo di sottorete, Amazon DocumentDB sceglie una sottorete e un indirizzo IP all'interno di tale sottorete da associare all'istanza principale del cluster. Il database utilizza la zona di disponibilità che

contiene la sottorete. DocumentDB assegna sempre un indirizzo IP da una sottorete che dispone di spazio libero per gli indirizzi IP.

Le sottoreti di un gruppo di sottoreti sono pubbliche o private, a seconda della configurazione impostata per gli elenchi di controllo degli accessi alla rete (ACL di rete) e le tabelle di routing. Affinché un cluster sia accessibile pubblicamente, tutte le sottoreti del relativo gruppo di sottoreti devono essere pubbliche. Se una sottorete associata a un cluster accessibile pubblicamente passa da pubblica a privata, può influire sulla disponibilità del cluster.

Per creare un gruppo di sottorete che supporti la modalità dual-stack, assicurati che a ogni sottorete aggiunta al gruppo di sottorete sia associato un blocco CIDR IPv6 (Internet Protocol version 6). Per ulteriori informazioni, consulta Indirizzamento IP di Amazon DocumentDB il supporto IPv6 per il tuo VPC nella Amazon Virtual Private Cloud User Guide.

Quando Amazon DocumentDB crea un cluster in un VPC, assegna un'interfaccia di rete al cluster utilizzando un indirizzo IP del tuo gruppo di sottorete. Usa sempre il nome DNS (Domain Name System) per connetterti al cluster perché l'indirizzo IP sottostante cambia durante il failover.

> **Note**
>
> Per ogni cluster eseguito in un VPC, assicurati di riservare almeno un indirizzo in ogni sottorete del gruppo di sottoreti affinché venga utilizzato da Amazon DocumentDB per le azioni di ripristino.

Sottoreti condivise

Puoi creare un cluster in un VPC condiviso.

Alcune considerazioni da tenere presente durante l'utilizzo di VPC condivisi:

- È possibile spostare un cluster da una sottorete VPC condivisa a una sottorete VPC non condivisa e viceversa.
- I partecipanti a un VPC condiviso devono creare un gruppo di sicurezza nel VPC per consentire loro di creare un cluster.
- I proprietari e i partecipanti a un VPC condiviso possono accedere al database utilizzando le query DocumentDB. Tuttavia, solo il creatore di una risorsa può effettuare chiamate API sulla risorsa.

Indirizzamento IP di Amazon DocumentDB

Gli indirizzi IP permettono alle risorse nel VPC di comunicare tra loro e con le risorse su Internet. Amazon DocumentDB supporta i protocolli di indirizzamento IPv4 e IPv6. Per impostazione predefinita, Amazon DocumentDB e Amazon VPC utilizzano il protocollo di indirizzamento IPv4. Non puoi disattivare questo comportamento. Quando crei un VPC, assicurati di specificare un blocco CIDR IPv4 (un intervallo di indirizzi IPv4 privati). Facoltativamente, puoi assegnare un blocco CIDR IPv6 al tuo VPC e alle tue sottoreti e assegnare gli indirizzi IPv6 di quel blocco ai cluster della tua sottorete.

> **Note**
>
> Dual-stack la modalità (indirizzamento IPv6) è supportata solo nelle versioni 4.0 e 5.0 di Amazon DocumentDB.

Il supporto del protocollo IPv6 espande il numero di indirizzi IP supportati. L'utilizzo del protocollo IPv6 consente di disporre di un numero sufficiente di indirizzi per adeguarsi alla futura espansione di Internet. Le risorse DocumentDB nuove ed esistenti possono utilizzare indirizzi IPv4 e IPv6 all'interno del tuo VPC. La configurazione, la protezione e la traduzione del traffico di rete tra i due protocolli utilizzati in diverse parti di un'applicazione può causare sovraccarico operativo. Puoi standardizzare il protocollo IPv6 per le risorse di Amazon DocumentDB per semplificare la configurazione di rete.

Argomenti

- Indirizzi IPv4
- Indirizzi IPv6
- Dual-stack modalità

Indirizzi IPv4

Quando crei un VPC, devi specificare un intervallo di indirizzi IPv4 per il VPC sotto forma di un blocco (CIDR), ad esempio `10.0.0.0/16`. Un gruppo di sottorete definisce l'intervallo di indirizzi IP in questo blocco CIDR che un cluster può utilizzare. Questi indirizzi IP possono essere privati o pubblici.

Un indirizzo IPv4 privato è un indirizzo IP non raggiungibile tramite Internet. Puoi utilizzare indirizzi IPv4 privati per la comunicazione tra il tuo cluster e altre risorse, come le istanze Amazon EC2, nello stesso VPC. Ogni cluster ha un indirizzo IP privato per la comunicazione nel VPC.

Un indirizzo IP pubblico è un indirizzo IPv4 raggiungibile tramite Internet. L'indirizzamento IP pubblico non è consentito per il cluster DocumentDB. Qualsiasi indirizzo IP pubblico deve essere risolto dal gateway Internet e da EC2 nella sottorete pubblica.

Per scoprire come creare un VPC con solo indirizzi IPv4 privati da utilizzare per uno scenario Amazon DocumentDB comune, consulta. Crea un IPv4-only VPC da utilizzare con un cluster DocumentDB

Indirizzi IPv6

Puoi anche scegliere di associare un blocco CIDR IPv6 al VPC e alle sottoreti e di assegnare gli indirizzi IPv6 del blocco alle risorse presenti nel VPC. Ogni indirizzo IPv6 è univoco a livello globale.

Il blocco CIDR IPv6 per il VPC è assegnato automaticamente dal pool di indirizzi IPv6 di Amazon. Non è possibile scegliere l'intervallo in modo autonomo.

Quando ti connetti a un indirizzo IPv6, assicurati che siano soddisfatte le seguenti condizioni:

- Il client è configurato in modo che sia consentito il traffico tra client e database su IPv6.
- I gruppi di sicurezza DocumentDB utilizzati dal cluster sono configurati correttamente in modo da consentire il traffico da client a database su IPv6.
- Lo stack del sistema operativo client consente il traffico sull'indirizzo IPv6 e i driver e le librerie del sistema operativo sono configurati per scegliere l'endpoint predefinito corretto del cluster (IPv4 o IPv6).

Per ulteriori informazioni sull'IPv6, consulta l'indirizzo IP nella Amazon Virtual Private Cloud User Guide.

Dual-stack modalità

Quando un cluster è in grado di comunicare tramite i protocolli di indirizzamento IPv4 e IPv6, viene eseguito in modalità dual-stack. Pertanto, le risorse possono comunicare con il cluster tramite IPv4, IPv6 o entrambi. DocumentDB disabilita l'accesso a Internet Gateway per gli endpoint IPv6 dei cluster privati in modalità dual-stack. DocumentDB esegue questa operazione per garantire che gli endpoint IPv6 siano privati e siano accessibili solo dall'interno del VPC.

Argomenti

- Dual-stack modalità e gruppi di sottorete
- Utilizzo di cluster in modalità dual-stack

- Modifica dei IPv4-only cluster per utilizzare la modalità dual-stack
- Dual-stack modalità: Regione e disponibilità della versione
- Limitazioni per i cluster di rete dual-stack

Dual-stack modalità e gruppi di sottorete

Per utilizzare la modalità dual-stack, assicurati che a ciascuna sottorete del gruppo di sottoreti che associ al cluster sia associato un blocco CIDR IPv6. È possibile creare un nuovo gruppo di sottorete o modificare un gruppo di sottorete esistente per soddisfare questo requisito. Dopo che un cluster è in modalità dual-stack, i client possono connettersi normalmente. Assicurati che i firewall di sicurezza dei client e i gruppi di sicurezza del cluster DocumentDB siano configurati accuratamente per consentire il traffico su IPv6. Per connettersi, i client utilizzano l'endpoint del cluster. Le applicazioni client possono specificare il protocollo preferito per la connessione a un database. In modalità dual-stack, il cluster rileva il protocollo di rete preferito dal client, IPv4 o IPv6, e utilizza tale protocollo per la connessione.

Se un gruppo di sottorete smette di supportare la modalità dual-stack a causa dell'eliminazione della sottorete o della dissociazione del CIDR, esiste il rischio di uno stato di rete incompatibile per i cluster associati al gruppo di sottorete. Inoltre, non è possibile utilizzare il gruppo di sottorete quando si crea un nuovo cluster in modalità dual-stack.

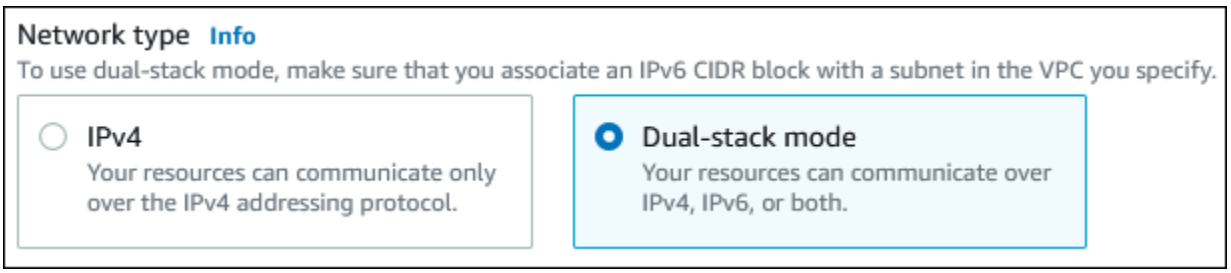
Per determinare se un gruppo di subnet supporta la modalità dual-stack utilizzando il Console di gestione AWS, visualizza il tipo di rete nella pagina dei dettagli del gruppo di sottorete. Per determinare se un gruppo di sottoreti supporta la modalità dual-stack utilizzando il AWS CLI, esegui il comando e visualizza nell'`describe-db-subnet-groups`output. `SupportedNetworkTypes`

Le repliche di lettura vengono trattate come cluster indipendenti e possono avere un tipo di rete diverso dal cluster primario. Se si modifica il tipo di rete del cluster primario di una replica di lettura, la replica di lettura non viene modificata. Quando si ripristina un cluster, è possibile ripristinarlo su qualsiasi tipo di rete supportato.

Utilizzo di cluster in modalità dual-stack

Quando si crea o si modifica un cluster, è possibile specificare la modalità dual-stack per consentire alle risorse di comunicare con il cluster tramite IPv4, IPv6 o entrambi.

Quando si utilizza Console di gestione AWS per creare o modificare un cluster, è possibile specificare la modalità dual-stack nella sezione Tipo di rete. L'immagine seguente mostra la sezione Tipo di rete nella console:



Quando utilizzi la AWS CLI per creare o modificare un cluster, imposta l'`--network-type`opzione per `DUAL` utilizzare la modalità dual-stack. Quando utilizzi l'API DocumentDB per creare o modificare un cluster, imposta il `NetworkType` parametro su per `DUAL` utilizzare la modalità dual-stack. Se la modalità dual-stack non è supportata dalla versione del motore DocumentDB o dal gruppo di sottorete specificato, viene restituito l'errore. `NetworkTypeNotSupported`

Per ulteriori informazioni sulla creazione di un cluster, vedere. Creazione di un cluster Amazon DocumentDB Per ulteriori informazioni sulla modifica di un cluster, vedereModifica di un cluster Amazon DocumentDB.

Per determinare se un cluster è in modalità dual-stack utilizzando la console, visualizza il tipo di rete nella scheda Connettività e sicurezza del cluster.

Modifica dei IPv4-only cluster per utilizzare la modalità dual-stack

È possibile modificare un IPv4-only cluster per utilizzare la modalità dual-stack. A tale scopo, modifica il tipo di rete del cluster.

Si consiglia di modificare il tipo di rete del cluster Amazon DocumentDB durante una finestra di manutenzione. È possibile impostare il tipo di rete manualmente utilizzando il `modify-db-cluster` comando.

Prima di modificare un cluster per utilizzare la modalità dual-stack, assicurati che il relativo gruppo di sottorete supporti la modalità dual-stack. Se il gruppo di sottorete associato al cluster non supporta la modalità dual-stack, specificate un gruppo di sottorete diverso che la supporti quando modificate il cluster. La modifica del gruppo di sottorete di un cluster può causare tempi di inattività.

Se modifichi il gruppo di sottorete di un cluster prima di modificare il cluster per utilizzare la modalità dual-stack, assicurati che il gruppo di sottorete sia valido per il cluster prima e dopo la modifica.

Esegui la `ModifyDBCluster` chiamata con il solo `NetworkType` parametro impostato su per `DUAL` cambiare la rete in modalità dual-stack. L'aggiunta di altri parametri `NetworkType` nella stessa chiamata API potrebbe causare tempi di inattività. Per modificare più parametri, assicurati

che la modifica del tipo di rete sia stata completata correttamente prima di inviare un'altra richiesta `ModifyDBCluster` con altri parametri.

Se non riesci a connetterti al cluster dopo la modifica, assicurati che i firewall di sicurezza del client e del database e le tabelle di routing siano configurati accuratamente per consentire il traffico verso il database sulla rete selezionata (IPv4 o IPv6). Potrebbe anche essere necessario modificare i parametri del sistema operativo, le librerie o i driver per connettersi mediante un indirizzo IPv6.

Per modificare un cluster per utilizzare la modalità dual-stack IPv4-only

1. Modifica un gruppo di sottorete per supportare la modalità dual-stack o crea un gruppo di sottorete che supporti la modalità dual-stack:

   a. Associazione di un blocco CIDR IPv6 al VPC.

      Per istruzioni, consulta Aggiungere o rimuovere un blocco CIDR al tuo VPC nella Amazon VPC User Guide.

   b. Collega il blocco CIDR IPv6 a tutte le sottoreti del tuo gruppo di sottoreti.

      Per istruzioni, consulta Aggiungere o rimuovere un blocco CIDR IPv6 alla tua sottorete nella Amazon Virtual Private Cloud User Guide.

   c. Verifica che il gruppo di sottorete supporti la modalità dual-stack.

      Se stai utilizzando il Console di gestione AWS, seleziona il gruppo di sottorete e assicurati che il valore Tipi di rete supportati sia Dual.

      Se stai usando il AWS CLI, esegui il `describe-db-subnet-groups` comando e assicurati che il `SupportedNetworkType` valore per il cluster sia`Dual`.

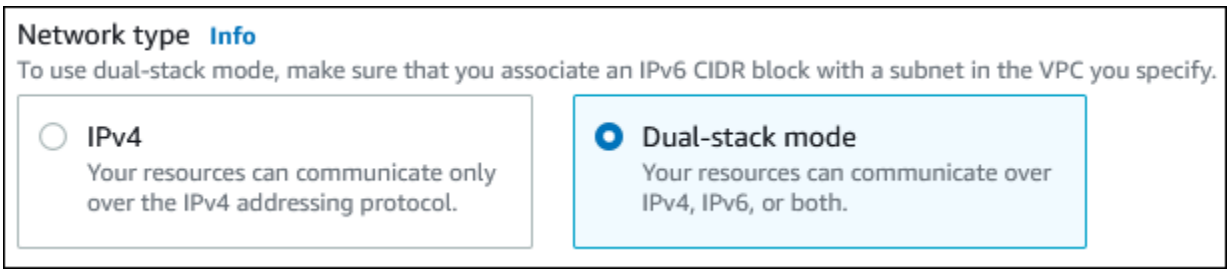
2. Modifica il gruppo di sicurezza associato al cluster per consentire le connessioni IPv6 al database o crea un nuovo gruppo di sicurezza che consenta le connessioni IPv6.

   Per istruzioni, consulta le regole dei gruppi di sicurezza nella Amazon Virtual Private Cloud User Guide.

3. Modifica il cluster per supportare la modalità dual-stack. A tale scopo, imposta il tipo di rete sulla modalità. Dual-stack

   In caso di utilizzo della console, accertati che le impostazioni seguenti siano corrette:

   - Tipo di rete: Dual-stack modalità



- Gruppo di sottorete: il gruppo di sottorete configurato in un passaggio precedente
- Gruppo di sicurezza: la sicurezza configurata in un passaggio precedente

Se stai utilizzando il AWS CLI, assicurati che le seguenti impostazioni siano corrette:

- `--network-type` — `dual`
- `--db-subnet-group-name`— Il gruppo di sottorete configurato in un passaggio precedente
- `--vpc-security-group-ids`— Il gruppo di sicurezza VPC che hai configurato in un passaggio precedente

Ad esempio:

```
aws docdb modify-db-cluster --db-cluster-identifier <cluster-name> --network-type
 "DUAL"
```

4. Verifica che il cluster supporti la modalità dual-stack.

   Se stai usando la console, scegli la scheda Connettività e sicurezza per il cluster e assicurati che il valore del tipo di rete sia Dual-stack mode.

   Se stai usando il AWS CLI, esegui il `describe-db-cluster` comando e assicurati che il `NetworkType` valore per il cluster sia`dual`.

   Esegui il `dig` comando sull'endpoint del cluster per identificare l'indirizzo IPv6 ad esso associato:

   ```
   dig <db-cluster-endpoint> AAAA
   ```

   Utilizza l'endpoint del cluster, non l'indirizzo IPv6, per connetterti al cluster.

Dual-stack modalità: Regione e disponibilità della versione

La disponibilità e il supporto delle funzionalità variano a seconda Regioni AWS.

Supporto di una regione

La modalità dual stack per Amazon DocumentDB è attualmente disponibile in tutte le versioni commerciali Regioni AWS.

Supporto per le versioni

Dual-stack la modalità è supportata nelle versioni 4.0, 5.0 e 8.0 di Amazon DocumentDB. Se non riesci ad accedere alla modalità dual-stack su nessuna di queste versioni, assicurati di eseguire l'ultima versione della patch del motore sul tuo cluster.

Limitazioni per i cluster di rete dual-stack

Le seguenti limitazioni si applicano ai cluster di rete dual-stack:

- I cluster non possono utilizzare esclusivamente il protocollo IPv6. Possono utilizzare esclusivamente il protocollo IPv4 oppure i protocolli IPv4 e IPv6 (modalità dual-stack).
- Amazon DocumentDB non supporta le sottoreti IPv6 native.
- I cluster che utilizzano la modalità dual-stack devono essere privati. Non possono essere accessibili pubblicamente.

Creazione di un cluster in un VPC

Le seguenti procedure consentono di creare un cluster in un VPC. Per utilizzare il VPC predefinito, puoi iniziare dal passaggio 2 e utilizzare il VPC e il gruppo di sottorete che sono già stati creati per te. Puoi anche creare VPC aggiuntivi, se necessario.

> **Note**
>
> Se desideri che il tuo cluster nel VPC sia accessibile al pubblico, devi aggiornare le informazioni DNS per il VPC abilitando gli attributi VPC e. `DNS hostnames` `DNS resolution` Per informazioni sull'aggiornamento del DNS per un'istanza VPC, consulta Visualizza e aggiorna gli attributi DNS per il tuo VPC nella Amazon Virtual Private Cloud User Guide.

Segui questi passaggi per creare un cluster in un VPC:

- Fase 1. Creazione di un VPC
- Fase 2: creazione di un gruppo di sottoreti
- Fase 3: creazione di un gruppo di sicurezza VPC
- Fase 4: Creare un cluster nel VPC

Fase 1. Creazione di un VPC

Crea un VPC con sottoreti in almeno due zone di disponibilità. Queste sottoreti vengono utilizzate quando si crea un gruppo di sottoreti. Se disponi di un VPC di default, viene creata automaticamente una sottorete in ciascuna zona di disponibilità nella Regione AWS.

Per ulteriori informazioniCrea un IPv4-only VPC da utilizzare con un cluster DocumentDB, consulta o consulta Create a VPC nella Amazon Virtual Private Cloud User Guide.

Fase 2: creazione di un gruppo di sottoreti

Un gruppo di sottoreti è una raccolta di sottoreti (in genere private) che crei per un VPC e che poi designi per i tuoi cluster. Un gruppo di sottoreti consente di specificare un particolare VPC quando si creano cluster utilizzando l'API o DocumentDB. AWS CLI Se utilizzi il Console di gestione AWS, puoi semplicemente scegliere il VPC e le sottoreti che desideri utilizzare. Ogni gruppo di sottorete deve avere almeno una sottorete in almeno due zone di disponibilità in. Regione AWS Come procedura consigliata, ogni gruppo di sottorete deve avere almeno una sottorete per ogni zona di disponibilità in. Regione AWS

Affinché un cluster sia accessibile al pubblico, le sottoreti del gruppo di sottoreti devono disporre di un gateway Internet. Per ulteriori informazioni sui gateway Internet per le sottoreti, consulta Abilitare l'accesso a Internet per un VPC utilizzando un gateway Internet nella Amazon Virtual Private Cloud User Guide.

> **Note**
>
> Il gruppo di sottorete per una zona locale può avere una sola sottorete.

Quando crei un cluster in un VPC, puoi scegliere un gruppo di sottorete. Amazon DocumentDB sceglie una sottorete e un indirizzo IP (all'interno di tale sottorete) da associare al cluster. Se

non esistono gruppi di sottorete, Amazon DocumentDB crea un gruppo di sottorete predefinito al momento della creazione del cluster. DocumentDB crea e associa un'interfaccia di rete elastica al cluster a quell'indirizzo IP. Il cluster utilizza la zona di disponibilità che contiene la sottorete.

In questo passaggio, crei un gruppo di sottoreti e aggiungi le sottoreti che hai creato per il tuo VPC.

Come creare un gruppo di sottoreti

1. Accedi a e apri la console Amazon Console di gestione AWS DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).
3. Scegli Crea.
4. In Nome, digita il nome del tuo gruppo di sottorete.
5. In Descrizione, digita una descrizione per il tuo gruppo di sottorete.
6. Nella sezione Aggiungi sottoreti, per VPC, scegli il VPC predefinito o il VPC che hai creato. Quindi scegli le zone di disponibilità che includono le sottoreti dalle zone di disponibilità, quindi scegli le sottoreti dalle sottoreti.
7. Scegli Crea.

   Il nuovo gruppo di sottorete viene visualizzato nell'elenco dei gruppi di sottorete sulla console DocumentDB. È possibile scegliere il gruppo di sottoreti per visualizzare i dettagli, incluse tutte le sottoreti associate al gruppo, nel riquadro dei dettagli nella parte inferiore della finestra.

Fase 3: creazione di un gruppo di sicurezza VPC

Prima di creare il cluster, crea un gruppo di sicurezza VPC da associare ad esso. Se non crei un gruppo di sicurezza VPC, puoi utilizzare il gruppo di sicurezza predefinito quando crei il cluster. Per istruzioni su come creare un gruppo di sicurezza per il tuo clusterCrea un IPv4-only VPC da utilizzare con un cluster DocumentDB, consulta o consulta Controllare il traffico verso AWS le tue risorse utilizzando i gruppi di sicurezza nella Amazon Virtual Private Cloud User Guide.

Fase 4: Creare un cluster nel VPC

In questo passaggio, crei un cluster utilizzando il nome VPC, il gruppo di sottorete e il gruppo di sicurezza VPC che hai creato nei passaggi precedenti.

> **Note**
>
> Se desideri che il tuo cluster nel VPC sia accessibile al pubblico, devi abilitare gli attributi VPC e. `DNS hostnames` `DNS resolution` Per ulteriori informazioni, consulta Visualizzare e aggiornare gli attributi DNS per il tuo VPC nella Amazon Virtual Private Cloud User Guide.

Per dettagli su come creare un cluster, consulta. Creazione di un cluster Amazon DocumentDB

Quando richiesto nella sezione Connettività, inserisci il nome del VPC, il gruppo di sottorete e il gruppo di sicurezza VPC.

> **Note**
>
> L'aggiornamento dei VPC non è attualmente supportato per i cluster DocumentDBDocumentDB.

Accesso a un cluster Amazon DocumentDB in un VPC

Amazon DocumentDB supporta i seguenti scenari per l'accesso a un cluster in un VPC:

Argomenti

- Un cluster in un VPC a cui si accede da un'istanza Amazon EC2 nello stesso VPC
- Un cluster in un VPC a cui si accede da un'istanza Amazon EC2 in un VPC diverso

Un cluster in un VPC a cui si accede da un'istanza Amazon EC2 nello stesso VPC

Un uso comune di un cluster in un VPC è condividere dati con un server applicativo in esecuzione in un'istanza Amazon EC2 nello stesso VPC.

Il modo più semplice per gestire l'accesso tra istanze EC2 e cluster nella stessa VPC è di seguire la seguente procedura:

- Crea un gruppo di sicurezza VPC in cui far risiedere i tuoi cluster. Questo gruppo di sicurezza può essere utilizzato per limitare l'accesso ai cluster. Ad esempio, puoi creare una regola personalizzata per questo gruppo di sicurezza. Ciò potrebbe consentire l'accesso TCP utilizzando

la porta assegnata al cluster al momento della creazione e un indirizzo IP utilizzato per accedere al cluster per lo sviluppo o per altri scopi.

- Crea un gruppo di sicurezza VPC in cui si troveranno le istanze EC2 (server Web e client). Questo gruppo di sicurezza può, se necessario, consentire l'accesso all'istanza EC2 da Internet tramite la tabella di routing del VPC. Ad esempio, può impostare regole in questo gruppo di sicurezza per consentire l'accesso TCP all'istanza EC2 sulla porta 22.
- Crea regole personalizzate nel gruppo di sicurezza per i tuoi cluster che consentano le connessioni dal gruppo di sicurezza che hai creato per le tue istanze EC2. Queste regole potrebbero consentire a qualsiasi membro del gruppo di sicurezza di accedere ai cluster.

È disponibile una sottorete pubblica e privata aggiuntiva in una zona di disponibilità separata. Un gruppo di sottoreti DocumentDB richiede una sottorete in almeno due zone di disponibilità. La sottorete aggiuntiva semplifica il passaggio a una distribuzione di cluster Multi-AZ in futuro.

Per istruzioni su come creare un VPC con sottoreti pubbliche e private per questo scenario, consulta. Crea un IPv4-only VPC da utilizzare con un cluster DocumentDB

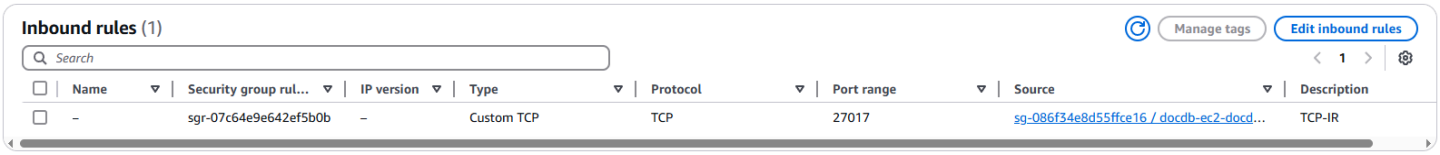
> **Tip**
>
> Puoi configurare automaticamente la connettività di rete tra un'istanza Amazon EC2 e un cluster DocumentDB quando crei il cluster. Per ulteriori informazioni, consulta Connetti Amazon EC2 automaticamente.

Per creare una regola in un gruppo di sicurezza VPC che consenta le connessioni da un altro gruppo di sicurezza, procedi come segue:

1. Accedi Console di gestione AWS e apri la console Amazon VPC all'indirizzo. https://console.aws.amazon.com/vpc
2. Nel pannello di navigazione, individua e seleziona Gruppi di sicurezza.
3. Scegli o crea un gruppo di sicurezza per il quale desideri concedere l'accesso ai membri di un altro gruppo di sicurezza. Questo è il gruppo di sicurezza che usi per i tuoi cluster. Seleziona la scheda Regole in entrata, quindi scegli Modifica regola.
4. Nella scheda Modifica regole in entrata, seleziona Aggiungi regola.
5. Per Tipo, scegli la voce che corrisponde alla porta utilizzata per creare il cluster, ad esempio Custom TCP.

6. Nel campo Origine, inizia a digitare l'ID del gruppo di sicurezza, che elenca i gruppi di sicurezza corrispondenti. Scegli il gruppo di sicurezza con i membri che desideri abbiano accesso alle risorse protette da questo gruppo di sicurezza. Nello scenario precedente, questo è il gruppo di sicurezza utilizzato per le istanze EC2.
7. Se necessario, ripeti i passaggi per il protocollo TCP creando una regola con All TCP come Tipo e il tuo gruppo di sicurezza nel campo Source. Se desideri usare il protocollo UDP, crea una regola con All UDP (Tutti i UDP) come Type (Tipo) e il gruppo di sicurezza nella casella Source (Origine).
8. Scegliere Salva regole.

Nella schermata seguente viene illustrata una regola in entrata con un gruppo di sicurezza per la relativa origine.



Per ulteriori informazioni sulla connessione a un cluster dalla tua istanza EC2, consulta. Connetti Amazon EC2 automaticamente

Un cluster in un VPC a cui si accede da un'istanza Amazon EC2 in un VPC diverso

Quando i tuoi cluster si trovano in un VPC diverso dall'istanza EC2 che stai utilizzando per accedervi, puoi utilizzare il peering VPC per accedere al cluster.

Una connessione di peering di VPC è una connessione di rete tra due VPC che consentono di instradare il traffico tra loro utilizzando degli indirizzi IP privati. Le risorse in uno qualsiasi dei VPC possono comunicare tra loro come se fossero nella stessa rete. Puoi creare una connessione peering VPC tra i tuoi VPC, con un VPC in un altro AWS account o con un VPC in un altro. Regione AWS Per ulteriori informazioni su VPC in peering, consulta Peering di VPC nella Guida per l'utente di Amazon Virtual Private Cloud.

Crea un IPv4-only VPC da utilizzare con un cluster DocumentDB

Uno scenario comune include un cluster in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC. Ad esempio, questo VPC potrebbe condividere dati con un servizio o un'applicazione in esecuzione nello stesso VPC. In questo argomento, crei il VPC per questo scenario.

Argomenti

- Fase 1: Creare un VPC con sottoreti private e pubbliche
- Passaggio 2: creare un gruppo di sicurezza VPC per un'applicazione pubblica
- Passaggio 3: creare un gruppo di sicurezza VPC per un cluster privato
- Fase 4: Creare un gruppo di sottorete
- Eliminazione di un VPC

Il cluster deve essere disponibile solo per l'applicazione e non per la rete Internet pubblica. Pertanto, occorre creare un VPC con sottoreti pubbliche e private. L'applicazione è ospitata nella sottorete pubblica, in modo che possa raggiungere la rete Internet pubblica. Il cluster è ospitato in una sottorete privata. L'applicazione può connettersi al cluster perché è ospitata all'interno dello stesso VPC. Tuttavia, il cluster non è disponibile per la rete Internet pubblica, il che garantisce una maggiore sicurezza.

La procedura descritta in questo argomento configura una sottorete pubblica e privata aggiuntiva in una zona di disponibilità separata. Queste sottoreti non vengono utilizzate dalla procedura. Un gruppo di sottoreti DocumentDB richiede una sottorete in almeno due zone di disponibilità. La sottorete aggiuntiva semplifica la configurazione di più di un'istanza di DocumentDB.

Questo argomento descrive la configurazione di un VPC per i cluster Amazon DocumentDB. Per ulteriori informazioni su Amazon VPC, consulta la Guida per l'utente di Amazon VPC.

> **Tip**
>
> È possibile configurare automaticamente la connettività di rete tra un'istanza Amazon EC2 e un cluster DocumentDB quando si crea il cluster. La configurazione di rete è simile a quella descritta in questo scenario. Per ulteriori informazioni, consulta Connetti Amazon EC2 automaticamente.

Fase 1: Creare un VPC con sottoreti private e pubbliche

Utilizza la procedura seguente per creare un VPC con sottoreti pubbliche e private.

Per creare un VPC e delle sottoreti

1. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.
2. Nell'angolo in alto a destra di Console di gestione AWS, scegli la regione in cui creare il tuo VPC. In questo esempio viene utilizzata la regione Stati Uniti occidentali (Oregon).

3. Nell'angolo in alto a sinistra, scegli VPC Dashboard (Pannello di controllo VPC). Per iniziare a creare un VPC, scegli Create VPC (Crea VPC).
4. In Resources to create (Risorse da creare) nell'area VPC settings (Impostazioni VPC), scegli VPC and more (VPC e altro).
5. In VPC settings (Impostazioni VPC) restanti, imposta i seguenti valori:
   - Generazione automatica dei tag nominativi — **`example`**
   - Blocco CIDR IPv4 — **`10.0.0.0/16`**
   - Blocco CIDR IPv6: nessun blocco CIDR IPv6
   - Locazione: impostazione predefinita
   - Numero di zone di disponibilità (AZ): 2
   - Personalizza aZs: mantieni i valori predefiniti
   - Numero di sottoreti pubbliche: 2
   - Numero di sottoreti private: 2
   - Personalizza le sottoreti, blocchi CIDR: mantieni i valori predefiniti
   - Gateway NAT ($): nessuno
   - Endpoint VPC: nessuno
   - Opzioni DNS: mantieni i valori predefiniti
6. Seleziona Crea VPC.

Passaggio 2: creare un gruppo di sicurezza VPC per un'applicazione pubblica

Quindi, crea un gruppo di sicurezza per l'accesso pubblico. Per connetterti alle istanze EC2 nel VPC, aggiungi regole in entrata al gruppo di sicurezza VPC. Queste consentono al traffico di connettersi da Internet.

Per creare un gruppo di sicurezza VPC

1. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.
2. Scegliere VPC Dashboard (Pannello di controllo VPC), Security Groups (Gruppi di sicurezza) e quindi Create security group (Crea gruppo di sicurezza).
3. Nella pagina Create security group (Crea gruppo di sicurezza) impostare questi valori:
   - Nome del gruppo di sicurezza: **`example-securitygroup`**

- Descrizione: **`Application security group`**
- VPC: scegli il VPC che hai creato in precedenza, ad esempio: vpc-example.

4. Aggiungere regole in entrata al gruppo di sicurezza

   a. Determina l'indirizzo IP da utilizzare per connettersi alle istanze EC2 nel VPC utilizzando Secure Shell (SSH). Per determinare il tuo indirizzo IP pubblico, in una finestra o scheda diversa del browser, puoi utilizzare il servizio all'indirizzo. https://checkip.amazonaws.com Un esempio di indirizzo IP è `203.0.113.25/32`.

      In molti casi, è possibile eseguire la connessione tramite un fornitore di servizi Internet (ISP) o con la protezione di un firewall senza un indirizzo IP statico. In tal caso, trova l'intervallo di indirizzi IP utilizzati dai computer client.

      > **Warning**
      >
      > Se utilizzi `0.0.0.0/0` per l'accesso SSH, consenti a tutti gli indirizzi IP di accedere alle istanze pubbliche utilizzando SSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, autorizza solo un determinato indirizzo IP o un intervallo di indirizzi per accedere alle istanze utilizzando SSH.

   b. Nella sezione Regole in entrata, scegliere Aggiungi regola.

   c. Imposta i seguenti valori per la nuova regola in entrata per consentire l'accesso SSH all'istanza Amazon EC2. Dopo averlo fatto, puoi connetterti alla tua istanza EC2 per installare l'applicazione e altre utilità. Ti connetti anche alla tua istanza EC2 per caricare contenuti per la tua applicazione.

      - Digita: **`SSH`**
      - Fonte: l'indirizzo o l'intervallo IP creato dalla Fase a, ad esempio: **`203.0.113.25/32`**

   d. Scegli Aggiungi regola.

   e. Imposta i seguenti valori per la nuova regola in entrata per consentire l'accesso HTTP alla tua applicazione:

      - Tipo: **`HTTP`**
      - Fonte: **`0.0.0.0/0`**

5. Per creare il gruppo di sicurezza, scegli Create security group (Crea gruppo di sicurezza).

Annota l'ID del gruppo di sicurezza perché ti servirà più avanti in un'altra procedura.

Passaggio 3: creare un gruppo di sicurezza VPC per un cluster privato

Per mantenere privato il tuo cluster, crea un secondo gruppo di sicurezza per l'accesso privato. Per connetterti ai cluster privati nel tuo VPC, aggiungi regole in entrata al tuo gruppo di sicurezza VPC che consentono il traffico solo dalla tua applicazione.

Per creare un gruppo di sicurezza VPC

1. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.
2. Scegliere VPC Dashboard (Pannello di controllo VPC), Security Groups (Gruppi di sicurezza) e quindi Create security group (Crea gruppo di sicurezza).
3. Nella pagina Create security group (Crea gruppo di sicurezza) impostare questi valori:
   - Nome del gruppo di sicurezza: **`example-securitygroup`**
   - Descrizione: **`Instance security group`**
   - VPC: scegli il VPC che hai creato in precedenza, ad esempio: vpc-example
4. Aggiungere regole in entrata al gruppo di sicurezza
   a. Nella sezione Regole in entrata, scegliere Aggiungi regola.
   b. Imposta i seguenti valori per la nuova regola in entrata per consentire il traffico di DocumentDB sulla porta 27017 dalla tua istanza Amazon EC2. Dopo averlo fatto, puoi connetterti dalla tua applicazione al tuo cluster. In questo modo, è possibile archiviare e recuperare i dati dall'applicazione al database.
      - Digitare: **`Custom TCP`**
      - Fonte: l'identificatore del gruppo di sicurezza dell'applicazione creato in precedenza in questo argomento, ad esempio: sg-9edd5cfb.
   c. Scegli Aggiungi regola.
   d. Imposta i seguenti valori per la nuova regola in entrata per consentire l'accesso HTTP all'applicazione:
      - Tipo: **`HTTP`**
      - Fonte: **`0.0.0.0/0`**
5. Per creare il gruppo di sicurezza, scegli Create security group (Crea gruppo di sicurezza).

Fase 4: Creare un gruppo di sottorete

Un gruppo di sottoreti è una raccolta di sottoreti che crei in un VPC e che poi designi per i tuoi cluster. Un gruppo di sottorete consente di specificare un particolare VPC durante la creazione dei cluster.

Come creare un gruppo di sottoreti

1. Identifica le sottoreti private per il database nel VPC.

   a. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.

   b. Scegli VPC Dashboard (Pannello di controllo VPC), quindi seleziona Subnets (Sottoreti).

   c. Annota gli ID di sottorete delle sottoreti create nel passaggio 1 denominati, ad esempio: example-subnet-private1-us-west-2a e example-subnet-private2-us-west-2b. Gli ID di sottorete sono necessari per creare il gruppo di sottorete.

2. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb

   Assicurati di connetterti alla console Amazon DocumentDB, non alla console Amazon VPC.

3. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).

4. Scegli Crea.

5. Nella pagina Crea gruppo di sottorete, imposta questi valori nella sezione dei dettagli del gruppo di sottorete:

   - Nome: **`example-db-subnet-group`**
   - Descrizione — **`Instance security group`**

6. Nella sezione Aggiungi sottoreti, imposta questi valori:

   - VPC: scegli il VPC che hai creato in precedenza, ad esempio: vpc-example
   - Zone di disponibilità: seleziona entrambe le zone di disponibilità create nel passaggio 1. Esempio: us-west-2a e us-west-2b
   - Sottoreti: scegli le sottoreti private che hai creato nel passaggio 1.

7. Scegli Crea.

Il nuovo gruppo di sottorete viene visualizzato nell'elenco dei gruppi di sottorete sulla console DocumentDB. È possibile scegliere il gruppo di subnet per visualizzare i dettagli nel riquadro dei dettagli. Questi dettagli includono tutte le sottoreti associate al gruppo.

> **Note**
>
> Se hai creato questo VPC per associarlo a un cluster DocumentDB, crea il cluster seguendo le istruzioni in. Creazione di un cluster Amazon DocumentDB

Eliminazione di un VPC

Puoi eliminare un VPC e le altre risorse utilizzate al suo interno, se non sono più necessarie.

> **Note**
>
> Se hai aggiunto risorse nel VPC che hai creato in questo argomento, potrebbe essere necessario eliminarle prima di poter eliminare il VPC. Ad esempio, queste risorse potrebbero includere istanze Amazon EC2 o cluster DocumentDB. Per ulteriori informazioni, consulta Eliminazione del VPC nella Guida per l'utente di Amazon VPC.

Per eliminare un VPC e le risorse correlate

1. Elimina il gruppo di sottorete:
    a. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
    b. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).
    c. Seleziona il gruppo di sottorete che desideri eliminare, ad esempio example-db-subnet-group.
    d. Scegliere Delete (Elimina) e quindi scegliere Delete (Elimina) nella finestra di conferma.
2. Annotare l'ID VPC:
    a. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.
    b. Scegli VPC Dashboard, quindi scegli I tuoi VPC.
    c. Nell'elenco, identifica il VPC che hai creato, ad esempio vpc-example.
    d. Prendi nota del valore ID VPC del VPC creato. L'ID VPC è richiesto nei passaggi successivi.
3. Eliminare i gruppi di sicurezza:
    a. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.

b. Scegliere VPC Dashboard (Pannello di controllo VPC) e quindi scegliere Security Groups (Gruppi di sicurezza).

c. Seleziona il gruppo di sicurezza per il cluster Amazon DocumentDB, ad esempio example-securitygroup.

d. Per Azioni, scegli Elimina gruppi di sicurezza, quindi scegli Elimina nella finestra di dialogo di conferma.

e. Tornando alla pagina Security Groups, seleziona il gruppo di sicurezza per l'istanza Amazon EC2, ad esempio example-securitygroup.

f. Per Azioni, scegli Elimina gruppi di sicurezza, quindi scegli Elimina nella finestra di dialogo di conferma.

4. Eliminare il VPC

a. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.

b. Scegli VPC Dashboard, quindi scegli I tuoi VPC.

c. Seleziona il VPC che desideri eliminare, ad esempio vpc-example.

d. Per Azioni, scegli Elimina VPC.

Nella pagina di conferma vengono visualizzate altre risorse associate al VPC che verranno eliminate, incluse le subnet associate.

e. Nella finestra di dialogo di conferma, inserisci**`delete`**, quindi scegli Elimina.

Crea un VPC dual-stack da utilizzare con un cluster DocumentDB

Uno scenario comune include un cluster in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC. Questo VPC condivide i dati con un'istanza Amazon EC2 pubblica in esecuzione nello stesso VPC. In questo argomento, crei il VPC per questo scenario.

Argomenti

- Fase 1: Creare un VPC con sottoreti private e pubbliche
- Passaggio 2: creare un gruppo di sicurezza VPC per un'istanza pubblica di Amazon EC2
- Passaggio 3: creare un gruppo di sicurezza VPC per un cluster privato
- Fase 4: Creare un gruppo di sottorete
- Passaggio 5: creare un'istanza Amazon EC2 in modalità dual-stack
- Passaggio 6: creazione di un cluster in modalità dual-stack

- Passaggio 7: connettiti alla tua istanza Amazon EC2 e al cluster DB
- Eliminare il VPC

In questa procedura, si crea il VPC per questo scenario che funziona con un database in esecuzione in modalità dual-stack. Dual-stack la modalità consente la connessione tramite il protocollo di indirizzamento IPv6. Per ulteriori informazioni sull'indirizzamento IP, consulta Indirizzamento IP di Amazon DocumentDB.

Dual-stack i cluster di rete sono supportati nella maggior parte delle regioni. Per ulteriori informazioni, consulta Dual-stack modalità: Regione e disponibilità della versione. Per esaminare le limitazioni della modalità dual-stack, consulta Limitazioni per i cluster di rete dual-stack.

Questo argomento e l' IPv4-only argomento creano le sottoreti pubbliche e private nello stesso VPC. Per informazioni sulla creazione del cluster Amazon DocumentDB in un VPC e dell'istanza Amazon EC2 in un VPC diverso, consulta. Accesso a un cluster Amazon DocumentDB in un VPC

Il cluster DocumentDB deve essere disponibile solo per la tua istanza Amazon EC2 e non per la rete Internet pubblica. Pertanto, occorre creare un VPC con sottoreti pubbliche e private. L'istanza EC2 è ospitata nella sottorete pubblica, in modo che possa raggiungere la rete Internet pubblica. Il cluster è ospitato in una sottorete privata. L'istanza EC2 può connettersi al cluster perché è ospitata all'interno dello stesso VPC. Tuttavia, il cluster non è disponibile per la rete Internet pubblica, il che garantisce una maggiore sicurezza.

Le procedure descritte in questo argomento configurano una sottorete pubblica e privata aggiuntiva in una zona di disponibilità separata. Queste sottoreti non vengono utilizzate dalla procedura. Un gruppo di sottoreti DocumentDB richiede una sottorete in almeno due zone di disponibilità. La sottorete aggiuntiva semplifica la configurazione di più di un'istanza di DocumentDB.

Per creare un cluster che utilizza la modalità dual-stack, specifica la Dual-stack modalità e per l'impostazione del tipo di rete. Puoi anche modificare un cluster con la stessa impostazione. Per ulteriori informazioni sulla creazione di un cluster, vedereCreazione di un cluster Amazon DocumentDB. Per ulteriori informazioni sulla modifica di un cluster di database, consultare Modifica di un cluster Amazon DocumentDB.

Questo argomento descrive la configurazione di un VPC per i cluster Amazon DocumentDB. Per ulteriori informazioni su Amazon VPC, consulta la Guida per l'utente di Amazon VPC.

Fase 1: Creare un VPC con sottoreti private e pubbliche

Utilizza la procedura seguente per creare un VPC con sottoreti pubbliche e private.

Per creare un VPC e delle sottoreti

1. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.
2. Nell'angolo in alto a destra di Console di gestione AWS, scegli la regione in cui creare il tuo VPC. In questo esempio viene utilizzata la regione Stati Uniti occidentali (Oregon).
3. Nell'angolo in alto a sinistra, scegli VPC Dashboard (Pannello di controllo VPC). Per iniziare a creare un VPC, scegli Create VPC (Crea VPC).
4. In Resources to create (Risorse da creare) nell'area VPC settings (Impostazioni VPC), scegli VPC and more (VPC e altro).
5. In VPC settings (Impostazioni VPC) restanti, imposta i seguenti valori:
   - Generazione automatica dei tag nominativi — **`example-dual-stack`**
   - Blocco CIDR IPv4 — **`10.0.0.0/16`**
   - Blocco CIDR IPv6 — blocco CIDR IPv6 Amazon-provided
   - Locazione: impostazione predefinita
   - Numero di zone di disponibilità (AZ): 2
   - Personalizza aZs: mantieni i valori predefiniti
   - Numero di sottoreti pubbliche: 2
   - Numero di sottoreti private: 2
   - Personalizza le sottoreti, blocchi CIDR: mantieni i valori predefiniti
   - Gateway NAT ($): nessuno
   - Gateway Internet solo in uscita: no
   - Endpoint VPC: nessuno
   - Opzioni DNS: mantieni i valori predefiniti
6. Seleziona Crea VPC.

Passaggio 2: creare un gruppo di sicurezza VPC per un'istanza pubblica di Amazon EC2

È possibile a questo punto aggiungere un gruppo di sicurezza per l'accesso pubblico. Per connettersi alle istanze EC2 nel VPC, aggiungi regole in entrata al gruppo di sicurezza VPC per consentire il traffico da Internet.

Per creare un gruppo di sicurezza VPC

1. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.
2. Scegliere VPC Dashboard (Pannello di controllo VPC), Security Groups (Gruppi di sicurezza) e quindi Create security group (Crea gruppo di sicurezza).
3. Nella pagina Create security group (Crea gruppo di sicurezza) impostare questi valori:
   - Nome del gruppo di sicurezza: **`example-dual-stack-securitygroup`**
   - Descrizione: **`Dual-stack security group`**
   - VPC: scegli il VPC che hai creato in precedenza, ad esempio: vpc-example-dual-stack.
4. Aggiungere regole in entrata al gruppo di sicurezza
   a. Determina l'indirizzo IP da utilizzare per connettersi alle istanze EC2 nel VPC utilizzando Secure Shell (SSH). Per determinare il tuo indirizzo IP pubblico, in una finestra o scheda diversa del browser, puoi utilizzare il servizio all'indirizzo. https://checkip.amazonaws.com

      Un esempio di intervallo di indirizzi IPv4 (Internet Protocol version 4) è. `203.0.113.25/32` Un esempio di intervalli di indirizzi Internet Protocol versione 6 (IPv6) è `2001:db8:1234:1a00::/64`.

      In molti casi, è possibile eseguire la connessione tramite un fornitore di servizi Internet (ISP) o con la protezione di un firewall senza un indirizzo IP statico. In tal caso, trova l'intervallo di indirizzi IP utilizzati dai computer client.

      > **Warning**
      >
      > Se utilizzi `0.0.0.0/0` per IPv4 o `::0` per IPv6, consenti a tutti gli indirizzi IP di accedere alle istanze pubbliche utilizzando SSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, è preferibile autorizzare l'accesso alle istanze solo a indirizzo IP o a un intervallo di indirizzi specifico.

   b. Nella sezione Regole in entrata, scegliere Aggiungi regola.
   c. Imposta i seguenti valori per la nuova regola in entrata per consentire l'accesso SSH all'istanza Amazon EC2. Dopo averlo fatto, puoi connetterti alla tua istanza EC2 per installare applicazioni o altre utilità. Specifica un indirizzo IP per poter accedere all'istanza EC2:
      - Tipo: **`SSH`**

- Fonte: l'indirizzo o l'intervallo IP creato dal passaggio a. Un esempio di intervallo di indirizzi IPv4 è **`203.0.113.25/3`** 2. Un esempio di intervallo di indirizzi IPv6 è. **`2001:DB8::/32`**

d. Scegli Aggiungi regola.

5. Per creare il gruppo di sicurezza, scegli Create security group (Crea gruppo di sicurezza).

   Annota l'ID del gruppo di sicurezza perché ti servirà più avanti in un'altra procedura.

Passaggio 3: creare un gruppo di sicurezza VPC per un cluster privato

Per mantenere privato il tuo cluster, crea un secondo gruppo di sicurezza per l'accesso privato. Per connetterti ai cluster privati nel tuo VPC, aggiungi regole in entrata al tuo gruppo di sicurezza VPC. Queste consentono il traffico solo dall'istanza Amazon EC2.

Per creare un gruppo di sicurezza VPC

1. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.
2. Scegliere VPC Dashboard (Pannello di controllo VPC), Security Groups (Gruppi di sicurezza) e quindi Create security group (Crea gruppo di sicurezza).
3. Nella pagina Create security group (Crea gruppo di sicurezza) impostare questi valori:
   - Nome del gruppo di sicurezza: **`example-dual-stack-cluster-securitygroup`**
   - Descrizione: **`Dual-stack cluster security group`**
   - VPC: scegli il VPC che hai creato in precedenza, ad esempio: vpc-example-dual-stack
4. Aggiungere regole in entrata al gruppo di sicurezza
   a. Nella sezione Regole in entrata, scegliere Aggiungi regola.
   b. Imposta i seguenti valori per la nuova regola in entrata per consentire il traffico DocumentDB sulla porta 27017 dalla tua istanza Amazon EC2. Dopo averlo fatto, puoi connetterti dalla tua istanza EC2 al tuo cluster. In questo modo, puoi inviare dati dalla tua istanza EC2 al tuo database.
      - Tipo: **`Custom TCP`**
      - Fonte: l'identificatore del gruppo di sicurezza EC2 creato in precedenza in questo argomento, ad esempio: sg-9edd5cfb.
5. Per creare il gruppo di sicurezza, scegli Create security group (Crea gruppo di sicurezza).

Fase 4: Creare un gruppo di sottorete

Un gruppo di sottoreti è una raccolta di sottoreti che crei in un VPC e che poi designi per i tuoi cluster. Utilizzando un gruppo di sottorete, è possibile specificare un particolare VPC durante la creazione dei cluster. Per creare un gruppo di sottoreti DUAL compatibile, tutte le sottoreti devono essere compatibili. DUAL Per essere compatibile con DUAL, a una sottorete deve essere associato un CIDR IPv6.

Come creare un gruppo di sottoreti

1. Identifica le sottoreti private per il database nel VPC.

   a. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.

   b. Scegli VPC Dashboard (Pannello di controllo VPC), quindi seleziona Subnets (Sottoreti).

   c. Annota gli ID di sottorete delle sottoreti create nel passaggio 1 denominati, ad esempio: example-dual-stack-subnet-private1-us-west-2a e example-dual-stack-subnet-private2-us-west-2b. Gli ID di sottorete sono necessari per creare il gruppo di sottorete.

2. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb

   Assicurati di connetterti alla console Amazon DocumentDB, non alla console Amazon VPC.

3. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).

4. Scegli Crea.

5. Nella pagina Crea gruppo di sottorete, imposta questi valori nella sezione dei dettagli del gruppo di sottorete:

   - Nome: **`example-dual-stack-cluster-subnet-group`**
   - Descrizione — **`Dual-stack cluster subnet group`**

6. Nella sezione Aggiungi sottoreti, imposta questi valori:

   - VPC: scegli il VPC che hai creato in precedenza, ad esempio: vpc-example-dual-stack
   - Zone di disponibilità: seleziona entrambe le zone di disponibilità create nel passaggio 1. Esempio: us-west-2a e us-west-2b
   - Sottoreti: scegli le sottoreti private che hai creato nel passaggio 1.

7. Scegli Crea.

Il nuovo gruppo di sottorete viene visualizzato nell'elenco dei gruppi di sottorete sulla console DocumentDB. È possibile scegliere il gruppo di subnet per visualizzare i dettagli nel riquadro dei dettagli. Questi dettagli includono tutte le sottoreti associate al gruppo.

Passaggio 5: creare un'istanza Amazon EC2 in modalità dual-stack

Per creare un'istanza Amazon EC2, segui le istruzioni in Avvia un'istanza EC2 utilizzando la procedura guidata di avvio dell'istanza nella console nella Amazon Elastic Compute Cloud User Guide.

Nella pagina Configure Instance Details (Configura i dettagli dell'istanza), imposta questi valori e lascia gli altri valori come predefiniti:

- Rete: scegli un VPC esistente con sottoreti pubbliche e private, ad esempio vpc-example-dual-stack-vpc (vpc-identifier) creato in. Fase 1: Creare un VPC con sottoreti private e pubbliche
- Sottorete: scegli una sottorete pubblica esistente, ad esempio subnet - identifier | example-dual-stack-subnet-public1-us-east-2a | us-east-2a creata in. Passaggio 2: creare un gruppo di sicurezza VPC per un'istanza pubblica di Amazon EC2
- Auto-assign IP pubblico: scegli Abilita.
- Auto-assign IP IPv6: scegli Abilita.
- Firewall (gruppi di sicurezza) Scegli Seleziona un gruppo di sicurezza esistente. —
- Gruppi di sicurezza comuni: scegli un gruppo di sicurezza esistente, ad esempio example-dual-stack-securitygroup creato in. Passaggio 2: creare un gruppo di sicurezza VPC per un'istanza pubblica di Amazon EC2 Assicurarsi che il gruppo di sicurezza scelto includa regole in ingresso per Secure Shell (SSH) e l'accesso HTTP.

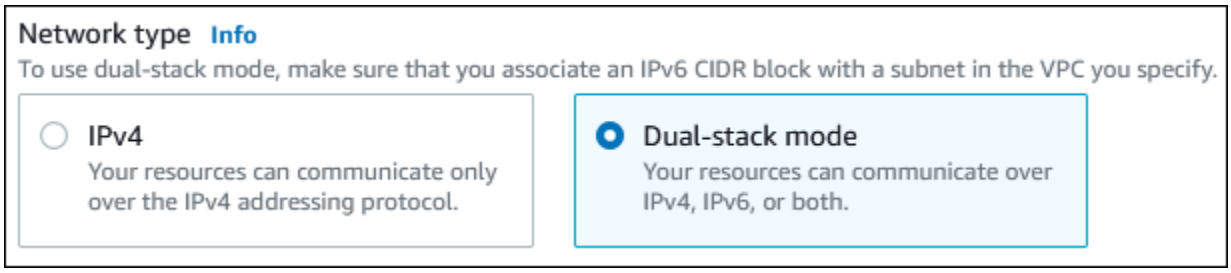
Passaggio 6: creazione di un cluster in modalità dual-stack

In questo passaggio, si crea un cluster DB che viene eseguito in modalità dual-stack. !!! Nota: questa sezione deve essere modificata una volta effettuati gli aggiornamenti IPv6 alla console!!!

Per creare un cluster in modalità dual-stack

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nell'angolo in alto a destra della console, scegli Regione AWS dove vuoi creare il cluster DocumentDB. Questo esempio utilizza la regione Stati Uniti orientali (Ohio).

3. Nel pannello di navigazione scegliere Cluster.
4. Nella pagina con l'elenco dei cluster, scegli Crea.
5. Nella pagina Crea cluster Amazon DocumentDB, assicurati che sia stata scelta l'opzione Instance-based cluster.
6. Nella sezione Connettività, in Tipo di rete, scegli la Dual-stack modalità.



7. Nella parte inferiore della pagina, attiva Mostra impostazioni avanzate.
8. Nella sezione Impostazioni di rete, imposta questi valori:

   - Cloud privato virtuale (VPC): scegli un VPC esistente con sottoreti pubbliche e private, come vpc-example-dual-stack (vpc-identifier) creato in. Fase 1: Creare un VPC con sottoreti private e pubbliche

     Il VPC deve avere sottoreti in diverse zone di disponibilità.
   - Gruppo di sottorete: scegli un gruppo di sottorete per il VPC, ad esempio example-dual-stack-cluster-subnet-group creato in. Fase 4: Creare un gruppo di sottorete
   - Accesso pubblico: scegli No.
   - Gruppo di sicurezza VPC (firewall): seleziona Scegli esistente.
   - Gruppi di sicurezza VPC esistenti: scegli un gruppo di sicurezza VPC esistente configurato per l'accesso privato, ad esempio example-dual-stack-cluster-securitygroup creato in. Passaggio 3: creare un gruppo di sicurezza VPC per un cluster privato

     Rimuovere gli altri gruppi di sicurezza, come quello predefinito, selezionando la X associata a esso.
   - Zona di disponibilità: scegli la zona di disponibilità creata nel passaggio 1. Esempio: us-west-2a.

     Per evitare il traffico cross-AZ, assicurati che il cluster e l'istanza EC2 si trovino nella stessa zona di disponibilità.
9. Per le sezioni rimanenti, specifica le impostazioni del cluster. Per informazioni su ciascuna impostazione, consulta Creazione di un cluster Amazon DocumentDB.

Passaggio 7: connettiti alla tua istanza Amazon EC2 e al cluster DB

Dopo aver creato l'istanza Amazon EC2 e il cluster DocumentDB in modalità dual-stack, puoi connetterti a ciascuno di essi utilizzando il protocollo IPv6. Per connetterti a un'istanza EC2 utilizzando il protocollo IPv6, segui le istruzioni in Connetti la tua istanza EC2 nella Amazon Elastic Compute Cloud User Guide.

Per connetterti al tuo cluster DocumentDB dall'istanza EC2, segui le istruzioni contenute nell'Passaggio 5: installa MongoDB Shellargomento Connetti Amazon EC2 manualmente (e continua con i successivi passaggi 6 e 7 nella stessa procedura).

Eliminare il VPC

Puoi eliminare un VPC e le altre risorse utilizzate al suo interno, se non sono più necessarie.

> **Note**
>
> Se hai aggiunto risorse nel VPC che hai creato in questo argomento, potrebbe essere necessario eliminarle prima di poter eliminare il VPC. Ad esempio, queste risorse potrebbero includere istanze Amazon EC2 o cluster DocumentDB. Per ulteriori informazioni, consulta Eliminazione del VPC nella Guida per l'utente di Amazon VPC.

Per eliminare un VPC e le risorse correlate

1. Elimina il gruppo di sottorete:
    a. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
    b. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).
    c. Seleziona il gruppo di sottorete che desideri eliminare, ad esempio example-dual-stack-cluster-subnet-group.
    d. Scegliere Delete (Elimina) e quindi scegliere Delete (Elimina) nella finestra di conferma.
2. Annotare l'ID VPC:
    a. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.
    b. Scegli VPC Dashboard, quindi scegli I tuoi VPC.
    c. Nell'elenco, identifica il VPC che hai creato, ad esempio vpc-example-dual-stack.

d. Prendi nota del valore ID VPC del VPC creato. L'ID VPC è richiesto nei passaggi successivi.

3. Eliminare i gruppi di sicurezza:

   a. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.

   b. Scegliere VPC Dashboard (Pannello di controllo VPC) e quindi scegliere Security Groups (Gruppi di sicurezza).

   c. Seleziona il gruppo di sicurezza per il cluster Amazon DocumentDB, ad esempio example-dual-stack-securitygroup.

   d. Per Azioni, scegli Elimina gruppi di sicurezza, quindi scegli Elimina nella finestra di dialogo di conferma.

   e. Tornando alla pagina Security Groups, seleziona il gruppo di sicurezza per l'istanza Amazon EC2, ad esempio example-securitygroup.

   f. Per Azioni, scegli Elimina gruppi di sicurezza, quindi scegli Elimina nella finestra di dialogo di conferma.

4. Elliminare il gateway NAT:

   a. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.

   b. Scegliere VPC Dashboard (Pannello di controllo VPC) e quindi scegliere Security Groups (Gruppi di sicurezza).

   c. Seleziona il gateway NAT del VPC creato. Utilizzare l'ID VPC per identificare il gateway NAT corretto.

   d. In Actions (Operazioni), scegli Delete NAT gateway (Elimina gateway NAT).

   e. Nella finestra di dialogo di conferma, inserisci**`delete`**, quindi scegli Elimina.

5. Eliminare il VPC

   a. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.

   b. Scegli VPC Dashboard, quindi scegli I tuoi VPC.

   c. Seleziona il VPC che desideri eliminare, ad esempio vpc-example-dual-stack.

   d. Per Azioni, scegli Elimina VPC.

      Nella pagina di conferma vengono visualizzate altre risorse associate al VPC che verranno eliminate, incluse le subnet associate.

   e. Nella finestra di dialogo di conferma, inserisci**`delete`**, quindi scegli Elimina.

6. Rilascia l'indirizzo IP elastico:

a. Apri la console EC2 all'indirizzo. https://console.aws.amazon.com/ec2
b. Scegliere EC2 Dashboard (Pannello di controllo EC2) e quindi scegliere Elastic IPs (IP elastici).
c. Seleziona l'indirizzo IP elastico da rilasciare.
d. In Actions (Operazioni), scegli Release Elastic IP addresses (Rilascia indirizzi IP elastici).
e. Nella finestra di dialogo di conferma, scegli Rilascia.

Backup e ripristino in Amazon DocumentDB

Amazon DocumentDB (con compatibilità MongoDB) esegue continuamente il backup dei dati su Amazon Simple Storage Service (Amazon S3) per 1—35 giorni in modo da poterli ripristinare rapidamente in qualsiasi momento entro il periodo di conservazione del backup. Amazon DocumentDB acquisisce anche istantanee automatiche dei tuoi dati come parte di questo processo di backup continuo.

> **Note**
>
> Si tratta di bucket Amazon S3 gestiti dal servizio e non avrai accesso ai file di backup. Per controllare i tuoi backup, consulta. Dumping, ripristino, importazione ed esportazione dei dati

Puoi anche conservare i dati di backup oltre il periodo di retention dei backup creando una snapshot dei dati del cluster. Il processo di backup non ha impatto sulle prestazioni del cluster.

Questa sezione illustra i casi d'uso delle funzionalità di backup in Amazon DocumentDB e mostra come gestire i backup per i cluster Amazon DocumentDB.

Argomenti

- Backup e ripristino: concetti
- Informazioni sull'utilizzo dello storage di backup
- Dumping, ripristino, importazione ed esportazione dei dati
- Considerazioni sulle istantanee del cluster
- Confronto tra istantanee automatiche e manuali
- Creazione manuale di un'istantanea del cluster
- Copia delle istantanee del cluster Amazon DocumentDB
- Condivisione delle istantanee del cluster Amazon DocumentDB
- Ripristino da un'istantanea del cluster
- Ripristino a un determinato punto nel tempo
- Eliminazione di un'istantanea del cluster

Backup e ripristino: concetti

Sostantivo	Description	API (verbi)

Periodo di conservazione dei backup	Un periodo di tempo compreso tra 1 e 35 giorni entro il quale puoi eseguire un ripristino a un punto temporale specifico.	`create-db-cluster`
		`modify-db-cluster`
		`restore-db-cluster-to-point-in-time`

Volume di archiviazione Amazon DocumentDB	Volume di storage a disponibilità elevata ed estremamente durevole che replica i dati in sei modi su tre zone di disponibilità. Un cluster Amazon DocumentDB è	`create-db-cluster` `delete-db-cluster`

Sostantivo	Description	API (verbi)
	estremamente resistente indipendentemente dal numero di istanze nel cluster.	

Backup window (Finestra di backup)	Periodo di tempo durante il giorno in cui vengono acquisite le snapshot automatiche.	`create-db-cluster`
		`describe-db-cluster`
		`modify-db-cluster`

Sostantivo	Description	API (verbi)

Snapshot automatica	Istantanee giornaliere che sono backup completi del cluster e vengono create automaticamente dal processo di backup continuo in Amazon DocumentDB.	`restore-db-cluster-from-snapshot`
		`describe-db-cluster-snapshot-attributes`
		`describe-db-cluster-snapshots`

Snapshot manuale	Snapshot create manualmente per conservare i backup completi di un cluster oltre il periodo di backup.	`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`

Informazioni sull'utilizzo dello storage di backup

Lo storage di backup di Amazon DocumentDB è costituito da backup continui entro il periodo di conservazione del backup e istantanee manuali al di fuori del periodo di conservazione. Per controllare l'utilizzo dello storage di backup, è possibile ridurre l'intervallo di retention dei backup e/

o rimuovere gli snapshot manuali obsoleti e non più necessari. Per informazioni generali sui backup di Amazon DocumentDB, consulta. Backup e ripristino in Amazon DocumentDB Per informazioni sui prezzi dello storage di backup di Amazon DocumentDB, consulta i prezzi di Amazon DocumentDB.

Per controllare i costi, è sufficiente monitorare la quantità di storage occupata da backup continui e snapshot manuali che permangono oltre il periodo di retention e, all'occorrenza, ridurre l'intervallo di retention dei backup nonché rimuovere gli snapshot manuali non più necessari.

Puoi utilizzare le CloudWatch metriche `TotalBackupStorageBilled` di Amazon e `BackupRetentionPeriodStorageUsed` per esaminare e monitorare la quantità di storage utilizzata dai backup di Amazon DocumentDB, come segue: `SnapshotStorageUsed`

- `BackupRetentionPeriodStorageUsed` rappresenta la quantità di storage di backup al momento utilizzata per l'archiviazione di backup continui. Il valore di questo parametro dipende dalle dimensioni del volume del cluster e dalla quantità di modifiche apportate durante il periodo di retention. Tuttavia, ai fini della fatturazione, il parametro non supera mai il volume di cluster cumulativo nel periodo di retention. Con un cluster di 100 GiB e un periodo di retention di due giorni, ad esempio, il valore massimo previsto per `BackRetentionPeriodStorageUsed` è di 200 GiB (100 GiB + 100 GiB).

- `SnapshotStorageUsed` rappresenta la quantità di storage di backup utilizzata per l'archiviazione di snapshot manuali oltre il periodo di retention dei backup. Gli snapshot manuali acquisiti nel periodo di retention non rientrano nel calcolo dello storage di backup. Analogamente, anche gli snapshot automatici non vengono inclusi nel novero dello storage di backup. Ogni snapshot corrisponde, per dimensioni, al volume del cluster al momento della sua acquisizione. Il valore di `SnapshotStorageUsed` dipende dal numero e dalle dimensioni degli snapshot conservati. Supponiamo, ad esempio, di disporre di uno snapshot al di fuori del periodo di retention, con il volume del cluster, al momento dell'acquisizione di tale snapshot, di 100 GiB. In questo caso, lo `SnapshotStorageUsed` corrisponde a 100 GiB.

- `TotalBackupStorageBilled` corrisponde alla somma di `BackupRetentionPeriodStorageUsed` e `SnapshotStorageUsed`, meno una quantità di storage di backup gratuito pari alla dimensione del volume del cluster per un giorno. Ad esempio, se la dimensione del cluster è di 100 GiB, hai un giorno di conservazione e hai un'istantanea al di fuori del periodo di conservazione, `TotalBackupStorageBilled` è 100 GiB (100 GiB + 100 GiB - 100 GiB).

- Queste metriche vengono calcolate indipendentemente per ogni cluster Amazon DocumentDB.

Puoi monitorare i tuoi cluster Amazon DocumentDB e creare report utilizzando CloudWatch le metriche tramite la console. CloudWatch Per ulteriori informazioni su come utilizzare le CloudWatch metriche, consulta. Monitoraggio e registrazione in Amazon DocumentDB

Dumping, ripristino, importazione ed esportazione dei dati

Puoi utilizzare le `mongoimport` utilità`mongodump`, `mongorestore``mongoexport`, e per spostare i dati in entrata e in uscita dal tuo cluster Amazon DocumentDB. In questa sezione viene descritto lo scopo di ognuno di questi strumenti e configurazioni per migliorare le prestazioni.

> **Requisiti di versione di MongoDB Database Tools**
>
> Usa MongoDB Database Tools fino alla versione 100.11.0 inclusa per Amazon DocumentDB. Per scaricarlo, consulta le versioni di MongoDB Database Tools sul sito Web di MongoDB.

Argomenti

- mongodump
- mongorestore
- mongoexport
- mongoimport
- Tutorial

`mongodump`

L'utilità `mongodump` crea un backup binario (BSON) di un database MongoDB. Lo `mongodump` strumento è il metodo preferito per scaricare i dati dalla distribuzione MongoDB di origine quando si cerca di ripristinarli nel cluster Amazon DocumentDB, grazie all'efficienza delle dimensioni ottenuta archiviando i dati in formato binario.

A seconda delle risorse disponibili sull'istanza o sulla macchina che stai utilizzando per eseguire il comando, puoi velocizzare il processo aumentando il numero di raccolte parallele scaricate rispetto

a quella mongodump predefinita 1 utilizzando l'opzione. `--numParallelCollections` Una buona regola empirica è iniziare con un lavoratore per vCPU sull'istanza principale del cluster Amazon DocumentDB.

Esempio di utilizzo

Di seguito è riportato un esempio di utilizzo dell'`mongodump`utilità nel cluster Amazon DocumentDB,. `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`

L'utilità `mongorestore` consente di ripristinare un backup binario (BSON) di un database creato con l'utilità mongodump. Puoi utilizzare l'opzione `--numInsertionWorkersPerCollection` per migliorare le prestazioni di ripristino aumentando il numero di worker per ogni raccolta durante il ripristino (il valore predefinito è 1). Una buona regola empirica è iniziare con un lavoratore per vCPU sull'istanza principale del cluster Amazon DocumentDB.

Esempio di utilizzo

Di seguito è riportato un esempio di utilizzo dell'`mongorestore`utilità nel cluster Amazon DocumentDB,. `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

Lo `mongoexport` strumento esporta i dati in Amazon DocumentDB in formati di file JSON, CSV o TSV. Lo strumento `mongoexport` è il metodo preferito per esportare i dati che devono essere leggibili dall'uomo o dal computer.

> **Note**
>
> `mongoexport` non supporta direttamente le esportazioni parallele. Tuttavia, è possibile aumentare le prestazioni eseguendo più attività `mongoexport` contemporaneamente per raccolte diverse.

Esempio di utilizzo

Di seguito è riportato un esempio di utilizzo dello `mongoexport` strumento nel cluster Amazon DocumentDB,. `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

Lo `mongoimport` strumento importa il contenuto dei file JSON, CSV o TSV in un cluster Amazon DocumentDB. È possibile utilizzare il parametro `--numInsertionWorkers` per parallelizzare e velocizzare l'importazione (il valore predefinito è 1).

Esempio di utilizzo

Di seguito è riportato un esempio di utilizzo dello `mongoimport` strumento nel cluster Amazon DocumentDB,. `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

Il seguente tutorial descrive come utilizzare le `mongoimport` utilità`mongodump`, `mongorestore``mongoexport`, e per spostare i dati in entrata e in uscita da un cluster Amazon DocumentDB.

1. Prerequisiti: prima di iniziare, assicurati che il cluster Amazon DocumentDB sia predisposto e di avere accesso a un'istanza Amazon EC2 nello stesso VPC del cluster. Per ulteriori informazioni, consulta Connettiti tramite Amazon EC2.

   Per poter utilizzare gli strumenti di utilità Mongo, è necessario disporre del pacchetto mongodb-org-tools installato nell'istanza EC2, come indicato di seguito.

   ```
   sudo yum install mongodb-org-tools-4.0.18
   ```

   Poiché Amazon DocumentDB utilizza la crittografia Transport Layer Security (TLS) per impostazione predefinita, devi anche scaricare il file dell'autorità di certificazione (CA) di Amazon RDS per utilizzare mongo shell per la connessione, come segue.

   ```
   wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
   ```

2. Scarica dati di esempio: per questo tutorial, scaricherai alcuni dati di esempio che contengono informazioni sui ristoranti.

   ```
   wget https://raw.githubusercontent.com/ozlerhakan/mongodb-json-files/master/
   datasets/restaurant.json
   ```

3. Importa i dati di esempio in Amazon DocumentDB: poiché i dati sono in un formato JSON logico, utilizzerai l'`mongoimport`utilità per importare i dati nel tuo cluster Amazon DocumentDB.

   ```
   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. Scarica i dati con **mongodump**: ora che hai dei dati nel tuo cluster Amazon DocumentDB, puoi eseguire un dump binario di tali dati utilizzando l'utilità. `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. Elimina la **`restaurants`** raccolta: prima di ripristinare la `restaurants` raccolta nel `business` database, devi prima eliminare la raccolta già esistente in quel database, come segue.

```
use business
```

```
db.restaurants.drop()
```

6. Ripristina i dati con **`mongorestore`**: con il dump binario dei dati di Step 3, ora puoi utilizzare l'`mongorestore`utilità per ripristinare i tuoi dati nel tuo cluster Amazon DocumentDB.

```
mongorestore --ssl \
    --host="tutorialCluster.us-east-1.docdb.amazonaws.com:27017" \
    --numParallelCollections 4 \
    --username=<yourUsername> \
    --password=<yourPassword> \
    --sslCAFile global-bundle.pem restaurantDump.bson
```

7. Esporta i dati utilizzando **`mongoexport`**: per completare il tutorial, esporta i dati dal tuo cluster nel formato di un file JSON, non diverso dal file che hai importato nel passaggio 1.

```
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. Convalida: puoi verificare che l'output della Fase 5 produca lo stesso risultato della Fase 1 con i seguenti comandi.

```
wc -l restaurant.json
```

Output da questo comando:

```
2548 restaurant.json
```

```
wc -l restaurant2.json
```

Output da questo comando:

```
2548 restaurant2.json
```

Considerazioni sulle istantanee del cluster

Amazon DocumentDB crea istantanee automatiche giornaliere del cluster durante la finestra di backup del cluster. Amazon DocumentDB salva le istantanee automatiche del cluster in base al periodo di conservazione del backup specificato. Se necessario, puoi ripristinare il tuo cluster a uno specifico momento durante il periodo di retention dei backup. Le snapshot automatiche non vengono eseguite quando un'operazione di copia è in corso nella stessa regione dello stesso cluster.

Argomenti

- Storage di backup
- Backup window (Finestra di backup)
- Periodo di conservazione dei backup

- Copia la crittografia delle istantanee del cluster

Oltre alle snapshot del cluster automatiche, puoi creare manualmente anche una snapshot del cluster. È possibile copiare gli snapshot automatici e manuali. Per ulteriori informazioni, consultare Creazione manuale di un'istantanea del cluster e Copia delle istantanee del cluster Amazon DocumentDB.

> **Note**
>
> Il cluster deve essere in stato disponibile per poter acquisire una snapshot automatica. Non puoi condividere uno snapshot automatico del cluster di Amazon DocumentDB. Come soluzione alternativa, è possibile creare uno snapshot manuale copiando lo snapshot automatico e quindi condividere la copia. Per ulteriori informazioni sulla copia di uno snapshot, consulta Copia delle istantanee del cluster Amazon DocumentDB. Per ulteriori informazioni sul ripristino di un cluster da uno snapshot, consulta Ripristino da un'istantanea del cluster.

Storage di backup

Lo storage di backup di Amazon DocumentDB per ciascuno di essi Regione AWS è composto dallo storage di backup necessario per il periodo di conservazione dei backup, che include istantanee automatiche e manuali del cluster in quella regione. Il periodo di retention dei backup predefinito è 1 giorno. Per ulteriori informazioni sui prezzi dello storage di backup, consulta i prezzi di Amazon DocumentDB.

Quando elimini un cluster, tutte le relative snapshot automatiche vengono eliminate e non possono essere recuperate. Tuttavia, le istantanee manuali non vengono eliminate quando si elimina un cluster. Se scegli di fare in modo che Amazon DocumentDB crei un'istantanea finale (istantanea manuale) prima che il cluster venga eliminato, puoi utilizzare l'istantanea finale per ripristinare il cluster.

Per ulteriori informazioni su snapshot e storage, consulta Informazioni sull'utilizzo dello storage di backup.

Backup window (Finestra di backup)

Le snapshot automatiche vengono eseguite quotidianamente durante la finestra di backup scelta. Se la snapshot richiede più tempo rispetto a quello previsto per la finestra di backup, il processo di backup continua fino al completamento, anche se la finestra di backup è terminata. La finestra di backup non può sovrapporsi con la finestra di manutenzione settimanale per il cluster.

Se non specifichi una finestra di backup preferita al momento della creazione del cluster, Amazon DocumentDB assegna una finestra di backup predefinita di 30 minuti. Questa finestra viene scelta a caso da un blocco di 8 ore associato alla regione del cluster. Puoi modificare la finestra di backup preferita modificando il cluster. Per ulteriori informazioni, consulta Modifica di un cluster Amazon DocumentDB.

Nome della regione	Regione	Blocco di tempo UTC
Stati Uniti orientali (Ohio)	us-east-2	03:00-11:00
Stati Uniti orientali (Virginia settentrionale)	us-east-1	03:00-11:00
Stati Uniti occidentali (Oregon)	us-west-2	06:00-14:00
Africa (Città del Capo)	af-south-1	Dalle 03:00 alle 11:00
Asia Pacific (Hong Kong)	ap-east-1	06:00-14:00
Asia Pacifico (Hyderabad)	ap-south-2	DALLE 06:30 ALLE 14:30
Asia Pacifico (Malesia)	ap-southeast-5	13:00-21:00
Asia Pacifico (Mumbai)	ap-south-1	06:00-14:00
Asia Pacifico (Osaka-Locale)	ap-northeast-3	12:00-20:00
Asia Pacific (Seoul)	ap-northeast-2	13:00-21:00
Asia Pacifico (Singapore)	ap-southeast-1	14:00-22:00
Asia Pacifico (Sydney)	ap-southeast-2	12:00-20:00
Asia Pacifico (Giacarta)	ap-southeast-3	08:00-16:00

Nome della regione	Regione	Blocco di tempo UTC
Asia Pacifico (Melbourne)	ap-southeast-4	11:00-19:00
Asia Pacifico (Thailandia)	ap-southeast-7	15:00-23:00
Asia Pacifico (Tokyo)	ap-northeast-1	13:00-21:00
Canada (Centrale)	ca-central-1	03:00-11:00
Canada occidentale (Calgary)	ca-west-1	18:00-02:00
Cina (Pechino)	cn-north-1	06:00-14:00
China (Ningxia)	cn-northwest-1	06:00-14:00
Europa (Francoforte)	eu-central-1	21:00-05:00
Europa (Zurigo)	eu-central-2	02:00-10:00
Europa (Irlanda)	eu-west-1	22:00-06:00
Europe (London)	eu-west-2	22:00-06:00
Europe (Milan)	eu-south-1	02:00-10:00
Europa (Parigi)	eu-west-3	23:59-07:29
Europa (Spagna)	eu-south-2	DALLE 02:00 ALLE 10:00
Europa (Stoccolma)	eu-north-1	DALLE 04:00 ALLE 12:00
Messico (Centrale)	mx-central-1	03:00-11:00
Medio Oriente (Emirati Arabi Uniti)	me-central-1	DALLE 05:00 ALLE 13:00
Sud America (San Paolo)	sa-east-1	00:00-08:00
Israele (Tel Aviv)	il-central-1	04:00-12:00
AWS GovCloud (US-East)	us-gov-east-1	17:00-01:00

Nome della regione	Regione	Blocco di tempo UTC
AWS GovCloud (US-West)	us-gov-west-1	06:00-14:00

Periodo di conservazione dei backup

Il periodo di conservazione dei backup è il numero di giorni in cui un backup automatico viene conservato prima di essere eliminato automaticamente. Amazon DocumentDB supporta un periodo di conservazione dei backup compreso tra 1 e 35 giorni.

Puoi impostare il periodo di retention dei backup al momento della creazione di un cluster. Se non imposti esplicitamente il periodo di retention dei backup, verrà assegnato un periodo predefinito di un giorno al cluster. Dopo aver creato un cluster, puoi modificare il periodo di conservazione del backup modificando il cluster utilizzando il o il. Console di gestione AWS AWS CLI Per ulteriori informazioni, consulta Modifica di un cluster Amazon DocumentDB.

Copia la crittografia delle istantanee del cluster

La crittografia di cluster e snapshot si basa su una chiave di crittografia KMS. L'ID della chiave KMS è l'Amazon Resource Name (ARN), l'identificatore della chiave KMS o l'alias della chiave KMS per la chiave di crittografia KMS.

Si applicano le seguenti linee guida e limitazioni:

- La crittografia viene dedotta dal cluster durante la creazione di un'istantanea. Se il cluster è crittografato, l'istantanea di quel cluster viene crittografata con la stessa chiave KMS. Se il cluster non è crittografato, lo snapshot non è crittografato.
- Se copi uno snapshot crittografato del cluster dal tuo account Amazon Web Services, puoi specificare un valore per `KmsKeyId` crittografare la copia con una nuova chiave di crittografia KMS. Se non specifichi un valore per`KmsKeyId`, la copia dello snapshot del cluster viene crittografata con la stessa chiave KMS dello snapshot del cluster di origine.
- Se copi uno snapshot crittografato del cluster condiviso da un altro account Amazon Web Services, devi specificare un valore per. `KmsKeyId`
- Per copiare uno snapshot crittografato del cluster in un'altra regione Amazon Web Services, imposta `KmsKeyId` l'ID della chiave KMS che desideri utilizzare per crittografare la copia dello snapshot del cluster nella regione di destinazione. Le chiavi di crittografia KMS sono specifiche

della regione Amazon Web Services in cui vengono create e non puoi utilizzare chiavi di crittografia da una regione Amazon Web Services in un'altra regione Amazon Web Services.

- Se copi un'istantanea del cluster non crittografata e specifichi un valore per il `KmsKeyId` parametro, viene restituito un errore.

Confronto tra istantanee automatiche e manuali

Le seguenti sono le caratteristiche principali delle istantanee automatiche e manuali di Amazon DocumentDB (con compatibilità MongoDB).

Le istantanee automatiche di Amazon DocumentDB hanno le seguenti caratteristiche chiave:

- Denominazione automatica delle istantanee: i nomi automatici delle istantanee seguono lo schema `rds:<cluster-name>-yyyy-mm-dd-hh-mm` e `yyyy-mm-dd-hh-mm` rappresentano la data e l'ora di creazione dell'istantanea.
- Creato automaticamente in base a una pianificazione: quando si crea o si modifica un cluster, è possibile impostare il periodo di conservazione del backup su un valore intero compreso tra 1 e 35 giorni. Per impostazione predefinita, i nuovi cluster hanno un periodo di retention dei backup di un giorno. Il periodo di retention dei backup definisce il numero di giorni di conservazione delle snapshot automatiche prima di essere eliminate automaticamente. Non puoi disabilitare i backup automatici sui cluster Amazon DocumentDB.

  Oltre a impostare il periodo di retention dei backup, puoi anche impostare la finestra di backup, l'ora del giorno in cui vengono create le snapshot automatiche.
- Eliminazione delle istantanee automatiche: le istantanee automatiche vengono eliminate quando si elimina il cluster delle istantanee automatiche. Non puoi eliminare manualmente una snapshot automatica.
- Incrementale: durante il periodo di conservazione del backup, gli aggiornamenti del database vengono registrati in modo che vi sia una registrazione incrementale delle modifiche.
- Ripristino da un'istantanea automatica: è possibile eseguire il ripristino da un'istantanea automatica utilizzando o il. Console di gestione AWS AWS CLI Quando si esegue il ripristino da un'istantanea utilizzando il AWS CLI, è necessario aggiungere le istanze separatamente dopo che il cluster è disponibile.
- Condivisione: non è possibile condividere uno snapshot automatico del cluster Amazon DocumentDB. Come soluzione alternativa, è possibile creare uno snapshot manuale copiando lo snapshot automatico e quindi condividere la copia. Per ulteriori informazioni sulla copia di

uno snapshot, consulta Copia delle istantanee del cluster Amazon DocumentDB. Per ulteriori informazioni sul ripristino di un cluster da uno snapshot, consulta Ripristino da un'istantanea del cluster.

- È possibile eseguire il ripristino da qualsiasi momento all'interno del periodo di conservazione del backup: poiché gli aggiornamenti del database vengono registrati in modo incrementale, è possibile ripristinare il cluster in qualsiasi momento all'interno del periodo di conservazione del backup.

  Quando si esegue il ripristino da un'istantanea automatica o da un ripristino point-in-time utilizzando il AWS CLI, è necessario aggiungere le istanze separatamente dopo che il cluster è disponibile.

Le istantanee manuali di Amazon DocumentDB hanno le seguenti caratteristiche chiave:

- Creati su richiesta: gli snapshot manuali di Amazon DocumentDB vengono creati su richiesta utilizzando la console di gestione Amazon DocumentDB o. AWS CLI
- Eliminazione di un'istantanea manuale: un'istantanea manuale viene eliminata solo quando la elimini esplicitamente utilizzando la console Amazon DocumentDB o. AWS CLI Una snapshot manuale non viene eliminata quando elimini il cluster.
- Backup completi: quando viene scattata un'istantanea manuale, viene creato e archiviato un backup completo dei dati del cluster.
- Denominazione manuale delle istantanee: si specifica il nome manuale dello snapshot. Amazon DocumentDB non aggiunge un `datetime` timbro al nome, quindi è necessario aggiungere tali informazioni se si desidera includerle nel nome.
- Ripristino da un'istantanea manuale: puoi eseguire il ripristino da un'istantanea manuale utilizzando la console o il. AWS CLI Quando si esegue il ripristino da un'istantanea utilizzando il AWS CLI, è necessario aggiungere le istanze separatamente dopo che il cluster è disponibile.
- Quote di servizio: sei limitato a un massimo di 100 istantanee manuali per. Regione AWS
- Condivisione: è possibile condividere istantanee manuali del cluster, che possono essere copiate da persone autorizzate. Account AWSÈ possibile condividere snapshot manuali crittografati o non crittografati. Per ulteriori informazioni sulla copia di uno snapshot, consulta Copia delle istantanee del cluster Amazon DocumentDB.
- Si esegue il ripristino a quando è stata scattata l'istantanea manuale: quando si esegue il ripristino da un'istantanea manuale, si ripristina a quando è stata scattata l'istantanea manuale.

  Quando si esegue il ripristino da un'istantanea utilizzando il AWS CLI, è necessario aggiungere le istanze separatamente dopo che il cluster è disponibile.

Creazione manuale di un'istantanea del cluster

Puoi creare un'istantanea manuale utilizzando o. Console di gestione AWS AWS CLI La quantità di tempo necessaria per creare uno snapshot varia a seconda della dimensione dei database. Quando crei una snapshot, devi:

1. Identificare il cluster di cui eseguire il backup.
2. Dare un nome alla tua snapshot. In questo modo potrai eseguire il ripristino da qui in un secondo momento.

Using the Console di gestione AWS

Per creare un'istantanea manuale utilizzando il Console di gestione AWS, puoi seguire uno dei metodi seguenti.

1. Metodo 1:

    1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
    2. Nel pannello di navigazione, selezionare Snapshots (Snapshot).

        > **Tip**
        >
        > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

    3. Nella pagina Snapshots (Snapshot) scegli Create (Crea).
    4. Nella pagina Create cluster snapshot (Crea snapshot cluster):
        a. Identificatore del cluster: dall'elenco a discesa dei cluster, scegli il cluster di cui desideri creare un'istantanea.
        b. Identificatore istantanea: inserisci un nome per la tua istantanea.

            Vincoli per la denominazione di snapshot:

            - La lunghezza è di [1—255] lettere, numeri o trattini.
            - Il primo carattere deve essere una lettera.

- Non può terminare con un trattino o contenere due trattini consecutivi.
- Deve essere univoco per tutti i cluster (su Amazon RDS, Amazon Neptune e Amazon DocumentDB) per account, per regione. AWS

c. Scegli Create (Crea).

2. Metodo 2:

1. Accedi a e apri la console Amazon Console di gestione AWS DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.

> **Tip**
>
> Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nella pagina Clusters (Cluster) scegli il pulsante a sinistra del cluster di cui eseguire lo snapshot.
4. Dal menu Actions (Operazioni) scegli Take snapshot (Acquisisci snapshot).
5. Nella pagina Create cluster snapshot (Crea snapshot cluster):

a. Identificatore dell'istantanea: inserisci un nome per la tua istantanea.

Vincoli per la denominazione di snapshot:

- La lunghezza è di [1—63] lettere, numeri o trattini.
- Il primo carattere deve essere una lettera.
- Non può terminare con un trattino o contenere due trattini consecutivi.
- Deve essere univoco per tutti i cluster (su Amazon RDS, Amazon Neptune e Amazon DocumentDB) per account, per regione. AWS

b. Scegli Create (Crea).

Using the AWS CLI

Per creare un'istantanea del cluster utilizzando il AWS CLI, utilizza l'operazione con i seguenti parametri. `create-db-cluster-snapshot`

Parameters

- **`--db-cluster-identifier`**: obbligatorio. Il nome del cluster di cui si sta eseguendo la snapshot. Questo cluster deve esistere ed essere disponibile.
- **`--db-cluster-snapshot-identifier`**: obbligatorio. Il nome della snapshot manuale che stai creando.

Nell'esempio seguente viene creato uno snapshot denominato `sample-cluster-snapshot` per un cluster denominato `sample-cluster`.

Per Linux, macOS o Unix:

```
aws docdb create-db-cluster-snapshot \
    --db-cluster-identifier sample-cluster \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Per Windows:

```
aws docdb create-db-cluster-snapshot ^
    --db-cluster-identifier sample-cluster ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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"
    }
}
```

Copia delle istantanee del cluster Amazon DocumentDB

In Amazon DocumentDB, puoi copiare istantanee all'interno della stessa Regione AWS o in un'altra. Regione AWS Puoi anche copiare istantanee condivise sul tuo account nello stesso account Regione AWS o in un altro. Regione AWS Per ulteriori informazioni sugli snapshot di condivisione, consulta Condivisione delle istantanee del cluster Amazon DocumentDB.

> **Note**
>
> Amazon DocumentDB ti addebita in base alla quantità di dati di backup e istantanee che conservi e al periodo di tempo in cui li conservi. Per ulteriori informazioni sullo storage associato ai backup e alle istantanee di Amazon DocumentDB, consulta. Informazioni sull'utilizzo dello storage di backup Per informazioni sui prezzi dello storage Amazon DocumentDB, consulta i prezzi di Amazon DocumentDB.

Argomenti

- Copia di snapshot condivise
- Copiare istantanee tra Regioni AWS
- Limitazioni
- Gestione della crittografia
- Considerazioni sui gruppi di parametri
- Copiare un'istantanea del cluster

Copia di snapshot condivise

Puoi copiare istantanee condivise con te da altri account. AWS Se stai copiando un'istantanea crittografata condivisa da un altro AWS account, devi avere accesso alla chiave di AWS KMS crittografia utilizzata per crittografare l'istantanea. Per ulteriori informazioni, consulta Gestione della crittografia.

Copiare istantanee tra Regioni AWS

Quando si copia un'istantanea su un'istantanea diversa da Regione AWS quella di origine Regione AWS, ogni copia è un'istantanea completa. Una copia istantanea completa contiene tutti i dati e i metadati necessari per ripristinare il cluster Amazon DocumentDB.

Limitazioni

Di seguito sono riportate alcune limitazioni che si applicano quando si copiano le snapshot:

- Se elimini una snapshot origine prima che la snapshot target diventi disponibile, la copia della snapshot può non riuscire. Verifica che la snapshot target abbia lo stato di `AVAILABLE` prima di eliminare una snapshot origine.
- Puoi avere un massimo di cinque richieste di copia di snapshot in corso in una singola regione di destinazione per account
- A seconda del soggetto Regioni AWS coinvolto e della quantità di dati da copiare, il completamento di una copia istantanea interregionale può richiedere ore. In alcuni casi, potrebbe esserci un numero elevato di richieste di copia di snapshot tra regioni da una determinata Regione AWS di origine. In questi casi, Amazon DocumentDB potrebbe mettere in coda le nuove richieste di copia interregionali provenienti da quella fonte Regione AWS fino al completamento di alcune copie in corso. Nessuna informazione di progresso viene visualizzata sulle richieste di copia mentre sono in coda. Le informazioni sul progresso vengono visualizzate quando inizia la copia.

Gestione della crittografia

Puoi copiare una snapshot che è stata crittografata utilizzando una chiave di crittografia AWS KMS . Se la copia di una snapshot crittografata, la copia della snapshot deve anche essere crittografata. Se copi un'istantanea crittografata all'interno della stessa Regione AWS, puoi crittografarla con la stessa chiave di AWS KMS crittografia dell'istantanea originale oppure puoi specificare una chiave di crittografia diversa. AWS KMS Se copi un'istantanea crittografata in più regioni, non puoi utilizzare per la copia la stessa chiave di AWS KMS crittografia utilizzata per l'istantanea di origine, poiché le

chiavi lo sono. AWS KMS Region-specific È invece necessario specificare una AWS KMS chiave valida nella destinazione Regione AWS n.

La snapshot di origine resta crittografata nel processo di copia. Per ulteriori informazioni, consulta Protezione dei dati in Amazon DocumentDB.

> **Note**
>
> Per gli snapshot del cluster Amazon DocumentDB, non è possibile crittografare uno snapshot del cluster non crittografato quando si copia l'istantanea.

Considerazioni sui gruppi di parametri

Quando copi un'istantanea tra regioni, la copia non include il gruppo di parametri utilizzato dal cluster Amazon DocumentDB originale. Quando ripristini un'istantanea per creare un nuovo cluster, quel cluster ottiene il gruppo di parametri predefinito per il paese in cui Regione AWS è stato creato. Per assegnare al nuovo cluster gli stessi parametri dell'originale, devi fare quanto segue:

1. Nella destinazione Regione AWS, crea un gruppo di parametri del cluster Amazon DocumentDB con le stesse impostazioni del cluster originale. Se ne esiste già uno nel nuovo Regione AWS, puoi usare quello.
2. Dopo aver ripristinato l'istantanea nella destinazione Regione AWS, modifica il nuovo cluster Amazon DocumentDB e aggiungi il gruppo di parametri nuovo o esistente del passaggio precedente. Per ulteriori informazioni, consulta Modifica di un cluster Amazon DocumentDB.

Copiare un'istantanea del cluster

Puoi copiare un cluster Amazon DocumentDB utilizzando Console di gestione AWS o the AWS CLI, come segue.

Using the Console di gestione AWS

Per creare una copia di un'istantanea del cluster utilizzando il Console di gestione AWS, completa i seguenti passaggi. Questa procedura consente di copiare istantanee di cluster crittografate o non crittografate, nella stessa Regione AWS regione o in più regioni.

1. Accedi a e apri la Console di gestione AWS console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb

2. Nel pannello di navigazione, scegli Istantanee, quindi scegli il pulsante a sinistra dell'istantanea che desideri copiare.

> **Tip**
>
> Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nel menu Actions (Operazioni) scegliere Copy (Copia).
4. Nella pagina Crea copia dello snapshot del cluster risultante, completa la sezione Impostazioni.

   a. Regione di destinazione: facoltativa. Per copiare l'istantanea del cluster in un'altra Regione AWS, sceglietela come Regione Regione AWS di destinazione.

   b. Nuovo identificatore dello snapshot: inserisci un nome per la nuova istantanea.

      Vincoli per la denominazione di snapshot di destinazione:

      - Non può essere il nome di uno snapshot esistente.
      - La lunghezza è di [1—63] lettere, numeri o trattini.
      - Il primo carattere deve essere una lettera.
      - Non può terminare con un trattino o contenere due trattini consecutivi.
      - Deve essere univoco per tutti i cluster di Amazon RDS, Neptune e Amazon DocumentDB per regione. Account AWS

   c. Copia tag: per copiare i tag presenti nell'istantanea sorgente nella copia dell'istantanea, scegli Copia tag.

5. Completa la sezione. Encryption-at-rest

   a. Crittografia inattiva: se l'istantanea non è crittografata, queste opzioni non sono disponibili perché non è possibile creare una copia crittografata da un'istantanea non crittografata. Se l'istantanea è crittografata, è possibile modificare quella AWS KMS key utilizzata durante la crittografia a riposo.

      Per ulteriori informazioni sulla crittografia delle copie delle istantanee, vedere. Copia la crittografia delle istantanee del cluster

Per ulteriori informazioni sulla crittografia dei dati inattivi, consultare Crittografia dei dati di Amazon DocumentDB inattivi.

b. AWS KMS Chiave: dall'elenco a discesa, scegli una delle seguenti opzioni:

- (impostazione predefinita) aws/rds: il numero di conto e l'ID AWS KMS chiave sono elencati dopo questa opzione.
- <some-key-name>— Se hai creato una chiave, questa viene elencata e disponibile tra cui scegliere.
- Inserisci un ARN chiave: nella casella ARN, inserisci l'Amazon Resource Name (ARN) per la tua chiave. AWS KMS Il formato dell'ARN è `arn:aws:kms:<region>:<accountID>:key/<key-id>` .

6. Per eseguire una copia della snapshot selezionata, scegliere Copy snapshot (Copia snapshot). In alternativa, puoi scegliere Annulla per non creare una copia dell'istantanea.

Using the AWS CLI

Per creare una copia di un'istantanea del cluster non crittografata utilizzando AWS CLI, utilizzare l'`copy-db-cluster-snapshot`operazione con i seguenti parametri. Se stai copiando l'istantanea su un'altra Regione AWS, esegui il comando nel quale verrà copiata Regione AWS l'istantanea.

- **`--source-db-cluster-snapshot-identifier`**: obbligatorio. L'identificatore della snapshot del cluster di cui eseguire una copia. Una snapshot del cluster con questo nome deve esistere ed essere disponibile. Se stai copiando l'istantanea su un'altra Regione AWS o se stai copiando un'istantanea del cluster condiviso, questo identificatore deve essere nel formato ARN dello snapshot del cluster di origine. Questo parametro non distingue tra maiuscole e minuscole.
- **`--target-db-cluster-snapshot-identifier`**: obbligatorio. L'identificatore della nuova snapshot del cluster da creare dalla snapshot del cluster di origine. Questo parametro non distingue tra maiuscole e minuscole.

  Vincoli per la denominazione di snapshot di destinazione:

  - Non può essere il nome di uno snapshot esistente.
  - La lunghezza è di [1—63] lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.

  - Non può terminare con un trattino o contenere due trattini consecutivi.
  - Deve essere univoco per tutti i cluster di Amazon RDS, Neptune e Amazon DocumentDB per regione. Account AWS
- **`--source-region`**— Se stai copiando l'istantanea su un'altra Regione AWS, specifica da dove verrà copiata l'istantanea Regione AWS crittografata del cluster.

  Se stai copiando l'istantanea su un'altra Regione AWS e non lo specifichi`--source-region`, devi invece specificare l'opzione. `pre-signed-url` Il `pre-signed-url` valore deve essere un URL che contiene una richiesta firmata Signature Version 4 affinché l'`CopyDBClusterSnapshot`azione venga richiamata nell'origine da Regione AWS cui viene copiata l'istantanea del cluster. Per ulteriori informazioni su`pre-signed-url`, vedere. CopyDBClusterSnapshot
- **`--kms-key-id`**— L'identificatore della chiave KMS per la chiave da utilizzare per crittografare la copia dello snapshot del cluster.

  Se si copia uno snapshot crittografato del cluster su un altro Regione AWS, questo parametro è obbligatorio. È necessario specificare una chiave KMS per la destinazione. Regione AWS

  Se si copia un'istantanea crittografata del cluster nello stesso Regione AWS, il parametro AWS KMS key è facoltativo. La copia dello snapshot del cluster è crittografata con la stessa AWS KMS chiave dello snapshot del cluster di origine. Se desideri specificare una nuova chiave di AWS KMS crittografia da utilizzare per crittografare la copia, puoi farlo utilizzando questo parametro.
- **`--copy-tags`**— Facoltativo. I tag e i valori da copiare.

Per annullare un'operazione di copia una volta che è in corso, è possibile eliminare l'istantanea del cluster di destinazione identificata da `--target-db-cluster-snapshot-identifier` o `TargetDBClusterSnapshotIdentifier` mentre l'istantanea del cluster è in stato di copia.

Example

Esempio 1: copia un'istantanea non crittografata nella stessa regione

L' AWS CLI esempio seguente crea una copia di `sample-cluster-snapshot` named `sample-cluster-snapshot-copy` in Regione AWS come l'istantanea di origine. Quando viene creata la copia, tutti i tag della snapshot originale vengono copiati nella copia della snapshot.

Per Linux, macOS o 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
```

Per 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
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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

Esempio 2: copia un'istantanea non crittografata su Regioni AWS

L' AWS CLI esempio seguente crea una copia di`sample-cluster-snapshot`, che ha l'ARN. `arn:aws:rds:us-east-1:123456789012:cluster-snapshot:sample-cluster-snapshot` Questa copia è denominata Named `sample-cluster-snapshot-copy` e si trova nella cartella Regione AWS in cui viene eseguito il comando.

Per Linux, macOS o 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
```

Per 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
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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

Esempio 3: copia un'istantanea crittografata su Regioni AWS

L' AWS CLI esempio seguente crea una copia `sample-cluster-snapshot` dalla regione us-west-2 alla regione us-east-1. Questo comando viene richiamato nella regione us-east-1.

Per Linux, macOS o 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
```

Per 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
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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

Esempio 4: copia un'istantanea condivisa non crittografata su Regioni AWS

L' AWS CLI esempio seguente, account -123456789012, crea una copia di un'istantanea del cluster non crittografata `sample-cluster-snapshot` condivisa dall'account, dalla regione 999999999999 us-east-1 alla regione us-west-2. Questo comando viene richiamato nella regione us-west-2. Per ulteriori informazioni sugli snapshot di condivisione, consulta Condivisione di uno snapshot.

Per Linux, macOS o 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
```

Per 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
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
 "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

Esempio 5: copia un'istantanea condivisa crittografata su Regioni AWS

L' AWS CLI esempio seguente, account - `123456789012` crea una copia di un'istantanea crittografata del cluster `sample-cluster-snapshot` condivisa dall'account, dalla regione us-east-1 alla regione us-west-2. `999999999999` L'istantanea di destinazione è crittografata con una chiave KMS gestita dal cliente -. `arn:aws:kms:us-west-2:123456789012:key/6c1f3264-1797-472b-ba37-03011e682d28` Questo

comando viene chiamato nella regione us-west-2. Per ulteriori informazioni sugli snapshot di condivisione, consulta Condivisione di uno snapshot.

Per Linux, macOS o 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
```

Per 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
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
 "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**
>
> Per ulteriori informazioni sulla crittografia delle copie delle istantanee, vedere. Copia la crittografia delle istantanee del cluster
> Per ulteriori informazioni sulla crittografia dei dati inattivi, consultare Crittografia dei dati di Amazon DocumentDB inattivi.

Condivisione delle istantanee del cluster Amazon DocumentDB

Utilizzando Amazon DocumentDB, puoi condividere un'istantanea manuale del cluster nei seguenti modi:

- La condivisione di un'istantanea manuale del cluster, crittografata o non crittografata, consente agli AWS account autorizzati di copiare l'istantanea.
- La condivisione di un'istantanea manuale del cluster, crittografata o non crittografata, consente agli AWS account autorizzati di ripristinare direttamente un cluster dall'istantanea anziché prenderne una copia e ripristinarla da quella.

> **Note**
>
> Per condividere un'istantanea automatica del cluster, crea un'istantanea manuale del cluster copiando l'istantanea automatica, quindi condividi quella copia. Questo processo si applica anche alle risorse generate dal backup. AWS

Puoi condividere un'istantanea manuale con un massimo di 20 altre istantanee. Account AWS Puoi anche condividere uno snapshot manuale non crittografato come pubblico, per renderlo disponibile a tutti gli account . Quando si condivide uno snapshot come pubblico, assicurarsi che non siano incluse informazioni personali.

Quando condividi istantanee manuali con altri Account AWS e ripristini un cluster da un'istantanea condivisa utilizzando l'API Amazon DocumentDB AWS CLI o Amazon DocumentDB, devi specificare l'Amazon Resource Name (ARN) dello snapshot condiviso come identificatore dello snapshot.

Condivisione di uno snapshot crittografato

Per la condivisione di snapshot crittografati vigono le seguenti restrizioni:

- Non puoi condividere snapshot crittografati come pubblici.
- Non puoi condividere un'istantanea crittografata utilizzando la chiave di AWS KMS crittografia predefinita dell'account che ha condiviso l'istantanea.

Per condividere snapshot crittografati, attieniti alla seguente procedura.

1. Condividi la chiave di crittografia AWS Key Management Service (AWS KMS) utilizzata per crittografare l'istantanea con tutti gli account a cui desideri poter accedere all'istantanea.

   Puoi condividere le chiavi di AWS KMS crittografia con altri AWS account aggiungendo gli altri account alla politica delle AWS KMS chiavi. Per informazioni dettagliate sull'aggiornamento di una policy chiave, consulta Using Key Policies in AWS KMS nella AWS Key Management Service Developer Guide. Per un esempio di creazione di una policy delle chiavi, consulta Creazione di una policy IAM per abilitare la copia di una snapshot crittografata più avanti in questo argomento.
2. Usa AWS CLI, come mostrato di seguito, per condividere l'istantanea crittografata con gli altri account.

Consentire l'accesso a un AWS KMS chiave crittografica

Affinché un altro utente possa Account AWS copiare un'istantanea crittografata condivisa dal tuo account, l'account con cui condividi l'istantanea deve avere accesso alla AWS KMS chiave che ha crittografato l'istantanea. Per consentire a un altro account di accedere a una AWS KMS chiave, aggiorna la politica relativa alla AWS KMS chiave con l'ARN dell'account con cui stai condividendo come principale nella politica delle chiavi. AWS KMS Successivamente, autorizza l'operazione `kms:CreateGrant`.

Dopo aver concesso a un account l'accesso alla tua chiave di AWS KMS crittografia, per copiare l'istantanea crittografata, quell'account deve creare un utente AWS Identity and Access Management (IAM) se non ne ha già uno. Inoltre, tale account deve anche allegare a quell'utente IAM una policy

IAM che consenta all'utente di copiare un'istantanea crittografata utilizzando la tua AWS KMS chiave. L'account deve essere un utente IAM e non può essere un' Account AWS identità root a causa delle restrizioni AWS KMS di sicurezza.

Nel seguente esempio di politica delle chiavi, l'utente 123451234512 è il proprietario della chiave di crittografia. AWS KMS L'utente 123456789012 corrisponde all'account con cui è stata condivisa la chiave. Questa politica sulle chiavi aggiornata consente all'account di accedere alla chiave. AWS KMS A tale scopo, include l'ARN per l' Account AWS identità principale dell'utente 123456789012 come principale della policy e consente l'azione. `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}}
        }
    ]
}
```

Creazione di una policy IAM per abilitare la copia di una snapshot crittografata

Quando l'esterno Account AWS ha accesso alla tua AWS KMS chiave, il proprietario di quell'account può creare una policy per consentire a un utente IAM creato per l'account di copiare un'istantanea crittografata crittografata con quella chiave. AWS KMS

L'esempio seguente mostra una policy che può essere associata a un utente IAM per Account AWS 123456789012. La policy consente all'utente IAM di copiare un'istantanea condivisa dall'account 123451234512 che è stata crittografata con la chiave nella regione us-west-2. 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
                }
            }
        }
    ]
}
```

Per dettagli sull'aggiornamento di una policy chiave, consulta Using Key Policies nella Developer Guide. AWS KMS AWS Key Management Service

Condivisione di uno snapshot

Puoi condividere un'istantanea manuale del cluster di Amazon DocumentDB (o una copia di un'istantanea automatica) utilizzando o: Console di gestione AWS AWS CLI

Using the Console di gestione AWS

Per condividere un'istantanea utilizzando il Console di gestione AWS, completa i seguenti passaggi:

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).
3. Selezionare lo snapshot manuale da condividere.
4. Nel menu a discesa Azioni, scegli Condividi.
5. Scegli una delle seguenti opzioni per la visibilità delle istantanee del database:

   - Se l'origine non è crittografata, scegli Pubblico per consentire a tutti gli AWS account di ripristinare un cluster dallo snapshot manuale. Oppure scegli Privato per consentire solo AWS agli account specificati di ripristinare un cluster dallo snapshot manuale.

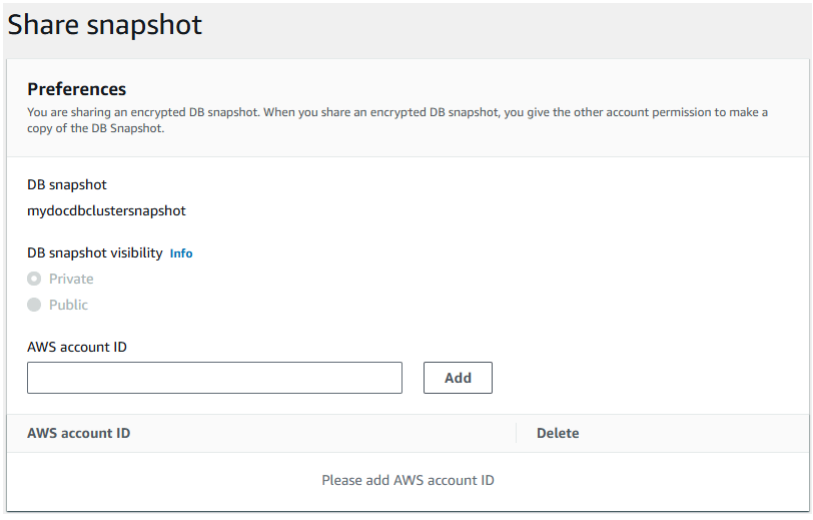
> **Warning**
>
> Se imposti la visibilità dello snapshot del database su Pubblico, tutti AWS gli account possono ripristinare un cluster dallo snapshot manuale e avere accesso ai tuoi dati. Non condividere le istantanee manuali del cluster che contengono informazioni private come pubbliche.

- Se l'origine è crittografata, l'opzione DB snapshot visibility (Visibilità snapshot DB) è impostata su Private (Privato), perché gli snapshot crittografati non possono essere condivisi come pubblici.

> **Note**
>
> Le istantanee che sono state crittografate con l'impostazione predefinita non AWS KMS key possono essere condivise.

6. Per ID AWS account, inserisci l' AWS identificativo dell'account a cui desideri consentire il ripristino di un cluster dallo snapshot manuale, quindi scegli Aggiungi. Ripeti l'operazione per includere identificatori di AWS account aggiuntivi, fino a 20 account. AWS

   Se commetti un errore quando aggiungi un identificativo di AWS account all'elenco degli account consentiti, puoi eliminarlo dall'elenco scegliendo Elimina a destra dell'identificativo di account errato AWS .



7. Dopo aver aggiunto gli identificatori per tutti gli AWS account a cui desideri consentire il ripristino dell'istantanea manuale, scegli Salva per salvare le modifiche.

Using the AWS CLI

Per condividere un'istantanea utilizzando il AWS CLI, utilizza l'operazione Amazon DocumentDB. `modify-db-snapshot-attribute` Utilizza il `--values-to-add` parametro per aggiungere un elenco degli ID degli utenti autorizzati Account AWS a ripristinare l'istantanea manuale.

L'esempio seguente consente a due Account AWS identificatori, 123451234512 e 123456789012, di ripristinare l'istantanea denominata. `manual-snapshot1` Inoltre, rimuove il valore dell'attributo `all` per contrassegnare lo snapshot come privato.

Per Linux, macOS o Unix:

```
aws docdb modify-db-cluster-snapshot-attribute \
    --db-cluster-snapshot-identifier sample-cluster-snapshot \
    --attribute-name restore \
    --values-to-add '["123451234512","123456789012"]'
```

Per Windows:

```
aws docdb modify-db-cluster-snapshot-attribute ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --attribute-name restore ^
    --values-to-add '["123451234512","123456789012"]'
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512",
                    "123456789012"
                ]
            }
        ]
    }
}
```

Per rimuovere un identificatore dall'elenco, utilizzate il parametro. Account AWS `--values-to-remove` L'esempio seguente impedisce all' Account AWS ID 123456789012 di ripristinare l'istantanea.

Per Linux, macOS o Unix:

```
aws docdb modify-db-cluster-snapshot-attribute \
    --db-cluster-snapshot-identifier sample-cluster-snapshot \
    --attribute-name restore \
    --values-to-remove '["123456789012"]'
```

Per Windows:

```
aws docdb modify-db-cluster-snapshot-attribute ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --attribute-name restore ^
    --values-to-remove '["123456789012"]'
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512"
                ]
            }
        ]
    }
}
```

Ripristino da un'istantanea del cluster

Amazon DocumentDB (con compatibilità con MongoDB) crea un'istantanea del cluster del volume di storage. Puoi creare un nuovo cluster ripristinandolo da una snapshot del cluster. Quando ripristini il cluster, devi fornire il nome della snapshot del cluster da cui effettuare il ripristino e il nome del nuovo

cluster creato dal ripristino. Non puoi eseguire il ripristino da una snapshot in un cluster esistente poiché al momento del ripristino viene creato un nuovo cluster.

Quando ripristini un cluster da una snapshot del cluster:

- Questa operazione ripristina solo il cluster, non le istanze del cluster. Devi utilizzare l'operazione `create-db-instance` per creare le istanze per il cluster ripristinato, specificandone l'identificatore in `--db-cluster-identifier`. Puoi creare le istanze solo dopo che il cluster è disponibile.
- Non puoi ripristinare una snapshot crittografata in un cluster non crittografato. Tuttavia, puoi ripristinare un'istantanea non crittografata in un cluster crittografato specificando la chiave. AWS KMS
- Per ripristinare un cluster da un'istantanea crittografata, è necessario avere accesso alla chiave. AWS KMS

> **Note**
>
> Non è possibile ripristinare un cluster 3.6 in un cluster 4.0, ma è possibile migrare da una versione del cluster a un'altra. Per ulteriori informazioni, consulta Migrazione e aggiornamento di Amazon DocumentDB.

Using the Console di gestione AWS

La procedura seguente mostra come ripristinare un cluster Amazon DocumentDB da uno snapshot del cluster utilizzando la Console di gestione Amazon DocumentDB.

1. Accedi a e apri la Console di gestione AWS console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel riquadro di navigazione scegli Snapshots (Snapshot) e seleziona il pulsante a sinistra della snapshot che desideri usare per ripristinare un cluster.

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu

(≡) nell'angolo in alto a sinistra della pagina.

3. Dal menu Actions (Operazioni) scegli Restore (Ripristina).
4. Nella pagina Restore snapshot (Ripristina snapshot) completa la sezione Configuration (Configurazione).

   a. Identificatore del cluster: il nome del nuovo cluster. Puoi accettare il nome fornito da Amazon DocumentDB o digitare il nome che preferisci. Il nome fornito da Amazon DocumentDB è nel formato `docdb-` più un timestamp UTC; ad esempio,. `docdb-yyyy-mm-dd-hh-mm-ss`

   b. Classe di istanza: la classe di istanza per il nuovo cluster. Puoi accettare la classe di istanza predefinita o scegliere una classe di istanza dall'elenco a discesa.

   c. Numero di istanze: il numero di istanze da creare con questo cluster. È possibile accettare il valore predefinito di 3 istanze (1 replica primaria read/write e 2 repliche di sola lettura) o scegliere il numero di istanze dall'elenco a discesa.

5. Per la configurazione dello storage in cluster, scegli un'opzione di archiviazione.

   Note

   La configurazione I/O-Optimized dello storage di Amazon DocumentDB è disponibile solo nelle versioni del motore Amazon DocumentDB 5.0 e 8.0.

6. Se si è soddisfatti della configurazione del cluster, scegliere Restore cluster (Ripristina cluster) e attendere il ripristino del cluster.
7. Se preferisci modificare alcune configurazioni, ad esempio specificando un Amazon VPC o un gruppo di sicurezza non predefinito, scegli Mostra impostazioni avanzate nella parte inferiore sinistra della pagina, quindi continua con i passaggi seguenti.

   a. Completa la sezione Network settings (Impostazioni di rete).

      - Virtual Private Cloud (VPC): accetta il VPC corrente o scegli un VPC dall'elenco a discesa.
      - Gruppo di sottorete: accetta il gruppo di `default` sottorete o scegline uno dall'elenco a discesa.
      - Gruppi di sicurezza VPC: accetta il gruppo `default (VPC)` di sicurezza o scegline uno dall'elenco.

b. Completa la sezione Cluster options (Opzioni cluster).

- Porta del database: accetta la porta predefinita o utilizza la freccia su o giù per impostare la porta che desideri utilizzare per le connessioni alle applicazioni. `27017`

c. Completa la sezione Encryption (Crittografia).

- Crittografia inattiva: se l'istantanea è crittografata, queste opzioni non sono disponibili. Se non è crittografata, puoi scegliere una delle seguenti opzioni:
  - Per crittografare tutti i dati del cluster, scegli Enable encryption-at-rest (Abilita la crittografia dei dati inattivi). Se scegli questa opzione, devi designare una chiave KMS.
  - Per non crittografare i dati del cluster, scegli Disable encryption-at-rest (Disabilita la crittografia dei dati inattivi). Se scegli questa opzione, hai terminato con la sezione relativa alla crittografia.
- AWS KMS Chiave: scegli una delle seguenti dall'elenco a discesa:
  - (impostazione predefinita) aws/rds: il numero di conto e l'ID AWS KMS chiave sono elencati dopo questa opzione.
  - Customer-managed chiave: questa opzione è disponibile solo se hai creato una chiave di crittografia IAM nella console AWS Identity and Access Management (IAM). Puoi scegliere la chiave per eseguire la crittografia del cluster.
  - Inserisci un ARN chiave: nella casella ARN, inserisci l'Amazon Resource Name (ARN) per la tua chiave. AWS KMS Il formato dell'ARN è `arn:aws:kms:<region>:<accountID>:key/<key-id>`.

d. Compila la sezione Log exports (Esportazioni di log).

- Seleziona i tipi di log su cui pubblicare CloudWatch: scegli una delle seguenti opzioni:
  - Abilitato: consente al cluster di esportare i log DDL in Amazon CloudWatch Logs.
  - Disabilitato: impedisce al cluster di esportare i log DDL in Amazon Logs. CloudWatch L'opzione predefinita è Disabled (Disabilitato).
- Ruolo IAM: dall'elenco, scegli RDS Service Linked Role.

e. Completa la sezione Tag .

- Aggiungi tag: nella casella Chiave, inserisci il nome del tag per il tuo cluster. Nella casella Value (Valore) immetti facoltativamente il valore del tag. I tag vengono utilizzati con le policy AWS Identity and Access Management (IAM) per gestire l'accesso

alle risorse di Amazon DocumentDB e per controllare quali azioni possono essere applicate alle risorse.

f. Completa la sezione Deletion protection (Protezione da eliminazione) .

- Abilita la protezione dalle eliminazioni: protegge il cluster dall'eliminazione accidentale. Quando questa opzione è abilitata, non puoi eliminare il cluster.

8. Scegli Restore cluster (Ripristina cluster).

Using the AWS CLI

Per ripristinare un cluster da un'istantanea utilizzando il AWS CLI, utilizza l'`restore-db-cluster-from-snapshot`operazione con i seguenti parametri. Per ulteriori informazioni, consulta RestoreDBClusterFromSnapshot.

- **`--db-cluster-identifier`**: obbligatorio. Il nome del cluster che viene creato dall'operazione. Prima di questa operazione non devono esistere cluster con questo nome.

  Vincoli per la denominazione del cluster:

  - La lunghezza è di [1—63] lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.
  - Deve essere univoco per tutti i cluster di Amazon RDS, Neptune e Amazon DocumentDB per regione. Account AWS
- **`--snapshot-identifier`**: obbligatorio. Il nome della snapshot utilizzata per eseguire il ripristino. Una snapshot con questo nome deve esistere ed essere disponibile.
- **`--engine`**: obbligatorio. Deve essere `docdb`.
- **`--storage-type standard | iopt1`**— Facoltativo. Default: `standard`.
- **`--kms-key-id`**— Facoltativo. L'ARN dell'identificatore AWS KMS chiave da utilizzare per il ripristino di un'istantanea crittografata o la crittografia di un cluster durante il ripristino da un'istantanea non crittografata. Se si fornisce l'ID della AWS KMS chiave, il cluster ripristinato viene crittografato con la chiave, indipendentemente dal fatto che lo snapshot sia crittografato o AWS KMS meno.

  Il formato di `--kms-key-id` è `arn:aws:kms:<region>:<accountID>:key/<key-id>`. Se non specifichi un valore per il parametro `--kms-key-id`, avviene quanto segue:

- Se lo snapshot in `--snapshot-identifier` è crittografato, il cluster ripristinato viene crittografato utilizzando la stessa AWS KMS chiave utilizzata per crittografare l'istantanea.
- Se la snapshot in `--snapshot-identifier` non è crittografata, il cluster ripristinato non è crittografato.

Per Linux, macOS o 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
```

Per 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
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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
    }
}
```

Dopo che lo stato del cluster è disponibile, crea almeno un'istanza per il cluster.

Per Linux, macOS o 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
```

Per 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
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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"
    }
}
```

Ripristino a un determinato punto nel tempo

Puoi ripristinare un cluster in qualsiasi momento compreso nel periodo di conservazione del backup del cluster utilizzando Console di gestione AWS or AWS Command Line Interface (AWS CLI).

> **Note**
>
> Non è possibile eseguire un ripristino point-in-time da un cluster 3.6 a un cluster 4.0, ma è possibile migrare da una versione del cluster a un'altra. Per ulteriori informazioni, consulta Migrazione e aggiornamento di Amazon DocumentDB.

Tieni presente quanto segue quando ripristini un cluster a un punto temporale specifico.

- Il nuovo cluster viene creato con la stessa configurazione del cluster di origine, ma utilizza il gruppo di parametri predefinito. Per impostare il gruppo di parametri del nuovo cluster su quello del cluster di origine, occorre modificare il cluster appena risulta disponibile. Per ulteriori informazioni sulla modifica di un cluster, consulta Modifica di un cluster Amazon DocumentDB.

Using the Console di gestione AWS

È possibile ripristinare un cluster in un point-in-time entro il periodo di retention dei backup completando quanto segue utilizzando la Console di gestione AWS.

1. Accedi a e apri la console Amazon Console di gestione AWS DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster. Dall'elenco di cluster scegliere il pulsante a sinistra del cluster che si desidera ripristinare.

> **Tip**
>
> Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nel menu Actions (Operazioni) scegliere Restore to point in time (Esegui ripristino point in time).
4. Completare la sezione Restore time (Ora ripristino), in cui si specificano data e ora per il ripristino.

   a. Data di ripristino: scegli o inserisci una data compresa tra il primo tempo di ripristino e l'ultimo orario di ripristino.

   b. Tempo di ripristino: scegli o inserisci l'ora, i minuti e i secondi compresi tra il primo ripristino e l'ultimo tempo di ripristino.
5. Completare la sezione Configurazione.

   a. Identificatore del cluster: accetta l'identificatore predefinito o inserisci un identificatore che preferisci.

      Vincoli per la denominazione del cluster:

      - La lunghezza è di [1—63] lettere, numeri o trattini.
      - Il primo carattere deve essere una lettera.
      - Non può terminare con un trattino o contenere due trattini consecutivi.
      - Deve essere univoco per tutti i cluster di Amazon RDS, Neptune e Amazon DocumentDB per regione. Account AWS

   b. Classe di istanza: dall'elenco a discesa, scegli la classe di istanza desiderata per le istanze del cluster.

   c. Numero di istanze: dall'elenco a discesa, scegli il numero di istanze da creare quando il cluster viene ripristinato.
6. Per la configurazione dello storage del cluster, scegli un'opzione di archiviazione.

> **Note**
>
> La configurazione I/O-Optimized dello storage di Amazon DocumentDB è disponibile solo nelle versioni del motore Amazon DocumentDB 5.0 e 8.0.

7. Opzionale. Per configurare le impostazioni di rete, le opzioni del cluster e abilitare le esportazioni dei log, scegliere Show advanced settings (Mostra impostazioni avanzate) e completare le sezioni seguenti. In alternativa, passare alla fase successiva.

   - Impostazioni di rete

     1. Virtual Private Cloud (VPC): dall'elenco a discesa, scegli il VPC che desideri utilizzare per questo cluster.
     2. Gruppo di sottorete: dall'elenco a discesa, scegli il gruppo di sottorete per questo cluster.
     3. Gruppi di sicurezza VPC: dall'elenco a discesa, scegli i gruppi di sicurezza VPC per questo cluster.

   - Cluster options (Opzioni cluster)

     1. Porta: accetta la porta predefinita (27017) o utilizza le frecce su e giù per impostare la porta per comunicare con questo cluster.

   - Log exports (Esportazioni log)

     1. Log di controllo: seleziona questa opzione per abilitare l'esportazione dei log di controllo in Amazon Logs. CloudWatch Se selezioni questa opzione, devi abilitare `audit_logs` nel gruppo personalizzato dei parametri del cluster. Per ulteriori informazioni, consulta Verifica degli eventi di Amazon DocumentDB.
     2. Log di profiler: seleziona questa opzione per abilitare l'esportazione dei log delle operazioni di profilazione in Amazon Logs. CloudWatch Se selezioni questa opzione, devi anche modificare i seguenti parametri nel gruppo personalizzato dei parametri del cluster:
        - `profiler`— Impostato su. `enabled`
        - `profiler_threshold_ms`— Impostato su un valore `[0-INT_MAX]` per impostare la soglia per le operazioni di profilazione.

- `profiler_sampling_rate`— Impostare un valore `[0.0-1.0]` per impostare la percentuale di operazioni lente da profilare.

  Per ulteriori informazioni, consulta Profilazione delle operazioni di Amazon DocumentDB.

3. Log del profiler: esporta i log del profiler in Amazon CloudWatch
4. Ruolo IAM: dall'elenco a discesa, scegli RDS Service Linked Role.

- Tag

1. Aggiungi tag: nella casella Chiave, inserisci il nome del tag per il tuo cluster. Nella casella Value (Valore) immetti facoltativamente il valore del tag. I tag vengono utilizzati con le policy AWS Identity and Access Management (IAM) per gestire l'accesso alle risorse di Amazon DocumentDB e per controllare quali azioni possono essere applicate alle risorse.

- Deletion protection (Protezione da eliminazione)

1. Abilita la protezione dalle eliminazioni: protegge il cluster dall'eliminazione accidentale. Quando questa opzione è abilitata, non puoi eliminare il cluster.

8. Per ripristinare il cluster, scegliere Create cluster (Crea cluster). In alternativa, è possibile scegliere Cancel (Annulla) per annullare l'operazione.

Using the AWS CLI

Per ripristinare un cluster a un punto temporale specifico con il periodo di retention dei backup della snapshot, utilizza l'operazione `restore-db-cluster-to-point-in-time` con i parametri seguenti.

- **`--db-cluster-identifier`**— Obbligatorio. Il nome del nuovo cluster da creare. Questo cluster non può esistere prima di questa operazione. Il valore del parametro deve soddisfare i seguenti vincoli.

  Vincoli per la denominazione del cluster:

  - La lunghezza è di [1—63] lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.
  - Deve essere univoco per tutti i cluster di Amazon RDS, Neptune e Amazon DocumentDB per regione. Account AWS

- **`--restore-to-time`**— La data e l'ora UTC in cui ripristinare il cluster. Ad esempio, `2018-06-07T23:45:00Z`.

  Vincoli temporali:

  - Deve essere prima dell'ultimo orario di ripristino per il cluster.
  - Deve essere specificato se il parametro `--use-latest-restorable-time` non viene fornito.
  - Non può essere specificato se il parametro `--use-latest-restorable-time` è `true`.
  - Non può essere specificato se il valore del parametro `--restore-type` è `copy-on-write`.
- **`--source-db-cluster-identifier`**— Il nome del cluster di origine da cui eseguire il ripristino. Questo cluster deve esistere ed essere disponibile.
- **`--use-latest-restorable-time`**oppure**`--no-use-latest-restorable-time`**: se eseguire il ripristino all'orario di backup ripristinabile più recente. Non può essere specificato se il parametro `--restore-to-time` viene fornito.
- **`--storage-type standard | iopt1`**— Opzionale. Default: `standard`.

L' AWS CLI operazione ripristina `restore-db-cluster-to-point-in-time` solo il cluster, non le istanze di quel cluster. È necessario utilizzare l'operazione `create-db-instance` per creare le istanze per il cluster ripristinato, specificandone l'identificatore in `--db-cluster-identifier`. Puoi creare le istanze solo dopo che l'operazione `restore-db-cluster-to-point-in-time` è terminata e il cluster ripristinato è disponibile.

Example

L'esempio seguente crea `sample-cluster-restored` dallo snapshot `sample-cluster-snapshot` all'ultimo orario di ripristino.

Per Linux, macOS o 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
```

Per 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

Nell'esempio seguente viene creato `sample-cluster-restored` dallo snapshot `sample-cluster-snapshot` delle 3:15 dell'11 dicembre 2018 (UTC), ovvero un orario incluso nel periodo di retention dei backup di `sample-cluster`.

Per Linux, macOS o 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
```

Per 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
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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
    }
}
```

Eliminazione di un'istantanea del cluster

Un'istantanea manuale è un backup completo che viene eliminato solo se lo elimini manualmente utilizzando il pulsante o. Console di gestione AWS AWS CLI Non è possibile eliminare manualmente una snapshot automatica, in quanto le snapshot automatiche vengono eliminate solo quando il periodo di retention delle snapshot scade o quando si elimina il rispettivo cluster.

Using the Console di gestione AWS

Per eliminare un'istantanea manuale del cluster utilizzando il Console di gestione AWS, completare i passaggi seguenti.

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione, selezionare Snapshots (Snapshot).

> **Tip**
>
> Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nell'elenco delle snapshot scegli il pulsante a sinistra della snapshot da eliminare. Il tipo di snapshot deve essere manuale.

   1. È possibile verificare che il tipo di snapshot sia manual (manuale) controllando se è elencato come `manual` o `automatic` sotto la colonna Type (Tipo).

4. Nel menu Actions (Operazioni) selezionare Delete (Elimina). Se l'opzione Delete (Elimina) non è disponibile, probabilmente hai scelto una snapshot automatica.
5. Nella schermata di conferma dell'eliminazione, scegli Delete (Elimina) per eliminare la snapshot. Per mantenere la snapshot, scegli Cancel (Annulla).

Using the AWS CLI

Un'istantanea manuale del cluster di Amazon DocumentDB è un backup completo che puoi eliminare manualmente utilizzando. AWS CLI Non puoi eliminare manualmente una snapshot automatica.

Per eliminare un'istantanea manuale del cluster utilizzando il AWS CLI, utilizza l'`delete-db-cluster-snapshot`operazione con i seguenti parametri.

Parameters

- **`--db-cluster-snapshot-identifier`**: obbligatorio. Il nome della snapshot manuale da eliminare.

L'esempio seguente elimina la snapshot del cluster `sample-cluster-snapshot`.

Per Linux, macOS o Unix:

```
aws docdb delete-db-cluster-snapshot \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Per Windows:

```
aws docdb delete-db-cluster-snapshot ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

L'output di questa operazione elenca i dettagli dello snapshot del cluster eliminato.

Monitoraggio e registrazione in Amazon DocumentDB

Il monitoraggio AWS dei tuoi servizi è una parte importante per mantenere i tuoi sistemi sani e funzionanti in modo ottimale. Raccogli i dati di monitoraggio da tutte le parti della AWS soluzione in modo da poter eseguire più facilmente il debug e correggere guasti o problemi di prestazioni. Prima di iniziare a monitorare le tue AWS soluzioni, considera le seguenti domande:

- Quali sono gli obiettivi del monitoraggio?
- Quali risorse verranno monitorate?
- Con quale frequenza eseguirai il monitoraggio di queste risorse?
- Quali strumenti di monitoraggio verranno utilizzati?
- Chi è il responsabile del monitoraggio?
- In caso di errore, a chi vanno inviate le notifiche e con quale modalità?

Per comprendere i tuoi modelli attuali relativi alle prestazioni, identificare le anomalie e formulare metodi per risolvere i problemi, è opportuno stabilire dei parametri di base per le prestazioni da applicare in momenti diversi e con diverse condizioni di carico. Mentre monitorate la AWS soluzione, archiviate i dati di monitoraggio cronologici per riferimenti futuri e per stabilire le vostre linee di base.

In generale, i valori accettabili per i parametri di prestazione dipendono dalla baseline e da cosa sta facendo l'applicazione. Indagare le variazioni della baseline coerenti o che rappresentano dei trend. Di seguito sono riportati alcuni suggerimenti su tipi di parametri specifici:

- Utilizzo elevato di CPU o RAM: valori elevati per l'utilizzo di CPU o RAM potrebbero essere appropriati, a condizione che siano in linea con gli obiettivi dell'applicazione (come la velocità effettiva o la concorrenza) e siano previsti.
- Consumo del volume di storage: analizza il consumo di storage (`VolumeBytesUsed`) se lo spazio utilizzato è costantemente pari o superiore all'85 percento dello spazio totale del volume di archiviazione. Verifica se puoi eliminare dati dal volume di archiviazione o archiviare dati su un sistema diverso per liberare spazio. Per ulteriori informazioni, consultare Storage Amazon DocumentDB e Quote Amazon DocumentDB.
- Traffico di rete: per quanto riguarda il traffico di rete, contatta l'amministratore di sistema per capire qual è il throughput previsto per la rete di dominio e la connessione Internet. Indaga il traffico di rete se il throughput è costantemente al di sotto del valore previsto.

- Connessioni al database: valuta la possibilità di limitare le connessioni al database se noti un numero elevato di connessioni utente insieme a una diminuzione delle prestazioni delle istanze e dei tempi di risposta. Il numero ideale di connessioni utente per l'istanza dipende dalla classe di istanza e dalla complessità delle operazioni eseguite.
- Metriche IOPS: i valori previsti per le metriche IOPS dipendono dalle specifiche del disco e dalla configurazione del server, quindi utilizza la tua baseline per sapere cosa è tipico. Verifica se i valori effettivi sono costantemente diversi rispetto a quelli di riferimento. Per prestazioni IOPS ottimali, verifica che il working set tipico possa essere caricato nella memoria per ridurre al minimo le operazioni di lettura e scrittura.

Amazon DocumentDB (con compatibilità con MongoDB) fornisce una serie di CloudWatch metriche Amazon che puoi monitorare per determinare lo stato e le prestazioni dei tuoi cluster e istanze Amazon DocumentDB. Puoi visualizzare le metriche di Amazon DocumentDB utilizzando vari strumenti, tra cui la console Amazon DocumentDB, l'API e Performance Insights. AWS CLI CloudWatch

Argomenti

- Monitoraggio dello stato di un cluster Amazon DocumentDB
- Monitoraggio dello stato di un'istanza Amazon DocumentDB
- Visualizzazione dei consigli di Amazon DocumentDB
- Utilizzo delle sottoscrizioni agli eventi di Amazon DocumentDB
- Monitoraggio di Amazon DocumentDB con CloudWatch
- Registrazione delle chiamate API di Amazon DocumentDB con AWS CloudTrail
- Profilazione delle operazioni di Amazon DocumentDB
- Monitoraggio con Performance Insights

Monitoraggio dello stato di un cluster Amazon DocumentDB

Lo stato di un cluster ne indica l'integrità. Puoi visualizzare lo stato di un cluster utilizzando la console Amazon DocumentDB o il AWS CLI `describe-db-clusters` comando.

Argomenti

- Valori di stato del cluster
- Monitoraggio dello stato di un cluster

Valori di stato del cluster

Nella tabella seguente sono elencati i valori validi per lo stato di un cluster.

Stato del cluster	Description
`active`	Il cluster è attivo. Questo stato si applica solo ai cluster elastici.
`available`	Il cluster è integro e disponibile. Questo stato si applica solo ai cluster basati su istanze.
`backing-up`	Il cluster è attualmente sottoposto a backup.
`creating`	Il cluster è in fase di creazione. L'accesso durante la creazione non è consentito.
`deleting`	Il cluster è in fase di eliminazione. L'accesso durante l'eliminazione non è consentito.
`failing-over`	È in corso un failover dall'istanza primaria a una replica di Amazon DocumentDB.
`inaccessible-encryption-credentials`	Non è AWS KMS possibile accedere alla chiave utilizzata per crittografare o decrittografare il cluster.
`maintenance`	È in corso l'applicazione di un aggiornamento di manutenzione al cluster. Questo stato viene utilizzato per la manutenzione a

Stato del cluster	Description
	livello di cluster che Amazon DocumentDB pianifica con largo anticipo.
`migrating`	Una snapshot del cluster è stata ripristinata in un cluster.
`migration-failed`	Una migrazione non è riuscita.
`modifying`	È in corso la modifica del cluster in seguito alla richiesta da parte di un cliente.
`renaming`	È in corso la ridenominazione del cluster in seguito alla richiesta da parte di un cliente.
`resetting-master-credentials`	È in corso il ripristino delle credenziali master del cluster in seguito alla richiesta da parte di un cliente.
`upgrading`	La versione del motore del cluster è in fase di aggiornam ento.

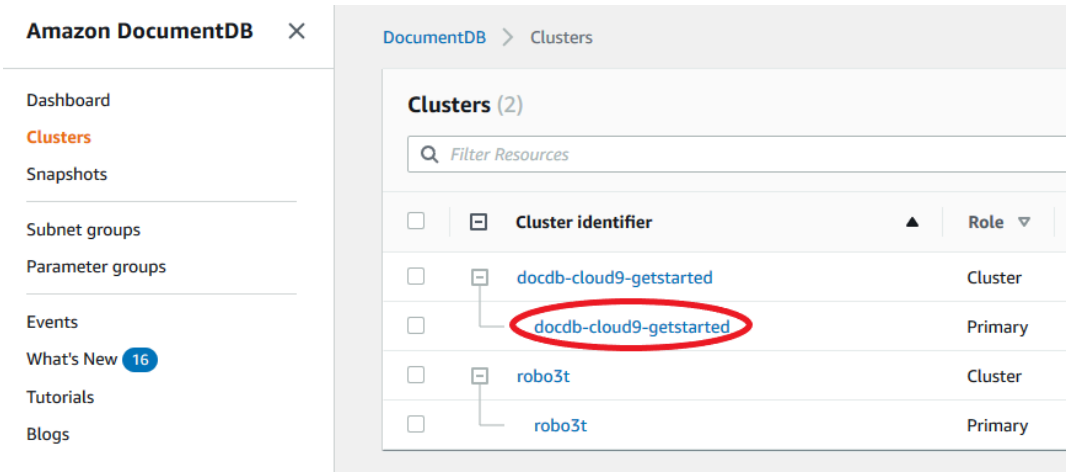
Monitoraggio dello stato di un cluster

Using the Console di gestione AWS

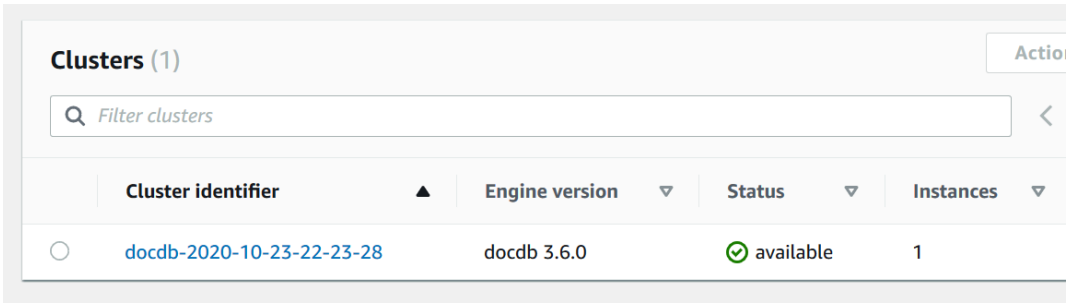
Quando si utilizza il Console di gestione AWS per determinare lo stato di un cluster, utilizzare la procedura seguente.

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.
2. Nel pannello di navigazione scegliere Cluster.

3. Nella casella di navigazione Clusters, vedrai la colonna Cluster identifier. Le tue istanze sono elencate in cluster, in modo simile alla schermata seguente.



4. Nella colonna Identificatore del cluster, trova il nome dell'istanza che ti interessa. Quindi, per trovare lo stato dell'istanza, leggi in quella riga fino alla colonna Status, come mostrato di seguito.



Using the AWS CLI

Quando si utilizza il AWS CLI per determinare lo stato di un cluster, utilizzare l'`describe-db-clusters`operazione. Il codice seguente individua lo stato del cluster `sample-cluster`.

Per Linux, macOS o Unix:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster  \
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Per Windows:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster  ^
```

```
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
[
    [
        "sample-cluster",
        "available"
    ]
]
```

Monitoraggio dello stato di un'istanza Amazon DocumentDB

Amazon DocumentDB fornisce informazioni sulla condizione corrente di ciascuna istanza configurata nel database.

Esistono tre tipi di stato che puoi visualizzare per un'istanza Amazon DocumentDB:

- Stato dell'istanza: questo stato è mostrato nella colonna Stato della tabella del cluster Console di gestione AWS e mostra lo stato attuale del ciclo di vita dell'istanza. I valori mostrati nella colonna Status derivano dal `Status` campo della risposta `DescribeDBCluster` API.
- Stato di integrità dell'istanza: questo stato è mostrato nella colonna Instance health della tabella del cluster in Console di gestione AWS e mostra se il motore di database, il componente responsabile della gestione e del recupero dei dati, è in esecuzione. I valori mostrati nella colonna Instance health si basano sulla metrica di CloudWatch `EngineUptime` sistema Amazon.
- Stato di manutenzione: questo stato è mostrato nella colonna Manutenzione della tabella del cluster in Console di gestione AWS e indica lo stato di qualsiasi evento di manutenzione che deve essere applicato a un'istanza. Lo stato di manutenzione è indipendente dallo stato dell'altra istanza ed è derivato dall'`PendingMaintenanceAction`API. Per ulteriori informazioni sullo stato della manutenzione, consulta Manutenzione di Amazon DocumentDB.

Argomenti

- Valori di stato delle istanze
- Monitoraggio dello stato dell'istanza utilizzando Console di gestione AWS or AWS CLI
- Valori dello stato di integrità dell'istanza
- Monitoraggio dello stato di integrità dell'istanza utilizzando Console di gestione AWS

Valori di stato delle istanze

Nella tabella seguente sono elencati i possibili valori di stato per le istanze e la modalità di fatturazione per ogni stato. Viene inoltre indicato se è prevista la fatturazione per l'istanza e lo storage, solo per lo storage oppure se la fatturazione non è prevista. Per tutti gli stati delle istanze, l'utilizzo del backup viene inserito in fattura.

stato dell'istanza	Fatturato	Description
`available`	Fatturato	L'istanza è integra e disponibile.
`backing-up`	Fatturato	L'istanza è attualmente sottoposta a backup.
`configuring-log-exports`	Fatturato	La pubblicazione dei file di CloudWatch log su Amazon Logs è abilitata o disabilitata per questa istanza.
`creating`	Non fatturato	L'istanza è in fase di creazione. Non è possibile accedere all'istanza mentre è in fase di creazione.
`deleting`	Non fatturato	L'istanza è in fase di eliminazione.
`failed`	Non fatturato	L'istanza è fallita e Amazon DocumentDB non è stato in grado di ripristinarla. Per ripristinare i dati, esegui un ripristino a un punto temporale specifico in base all'ultimo orario di ripristino dell'istanza.
`inaccessible-encryption-credentials`	Non fatturato	Non è stato possibile accedere alla AWS KMS chiave utilizzata per crittografare o decrittografare l'istanza.
`incompatible-network`	Non fatturato	Amazon DocumentDB sta tentando di eseguire un'azione di ripristino su un'istanza ma non è in grado di farlo perché il VPC si trova in uno stato che impedisce il completamento dell'azione. Questo stato può verificarsi se, ad esempio, tutti gli

stato dell’istanza	Fatturato	Description
		indirizzi IP disponibili in una sottorete erano in uso e Amazon DocumentDB non è stato in grado di ottenere un indirizzo IP per l'istanza.
`maintenance`	Fatturato	Amazon DocumentDB sta applicando un aggiornamento di manutenzione all'istanza. Questo stato viene utilizzato per la manutenzione a livello di istanza che Amazon DocumentDB pianifica con largo anticipo. Stiamo valutando nuovi modi per fornire altre operazioni di manutenzione ai clienti attraverso questo stato.
`modifying`	Fatturato	È in corso la modifica dell'istanza in seguito a una richiesta di eseguire tale operazione.
`rebooting`	Fatturato	L'istanza viene riavviata a causa di una richiesta o di un processo Amazon DocumentDB che richiede il riavvio dell'istanza.
`renaming`	Fatturato	È in corso la ridenominazione dell'istanza in seguito a una richiesta di eseguire tale operazione.
`resetting-master-credentials`	Fatturato	È in corso il ripristino delle credenziali master dell'istanza in seguito a una richiesta di eseguire tale operazione.
`restore-error`	Fatturato	L'istanza ha riscontrato un errore durante un tentativo ripristino a un punto temporale specifico o da una snapshot.
`starting`	Fatturato per storage	L'istanza è in fase di avvio.
`stopped`	Fatturato per storage	L'istanza è stata arrestata.

stato dell'istanza	Fatturato	Description
`stopping`	Fatturato per storage	L'istanza è in fase di arresto.
`storage-full`	Fatturato	L'istanza ha raggiunto la capacità di storage allocata. Si tratta di uno stato critico che richiede l'immediata risoluzione del problema; dimensionare la capacità di storage modificando l'istanza. Imposta gli CloudWatch allarmi di Amazon per avvisarti quando lo spazio di archiviazione si sta esaurendo, in modo da non incorrere in questa situazione.

Monitoraggio dello stato dell'istanza utilizzando Console di gestione AWS or AWS CLI

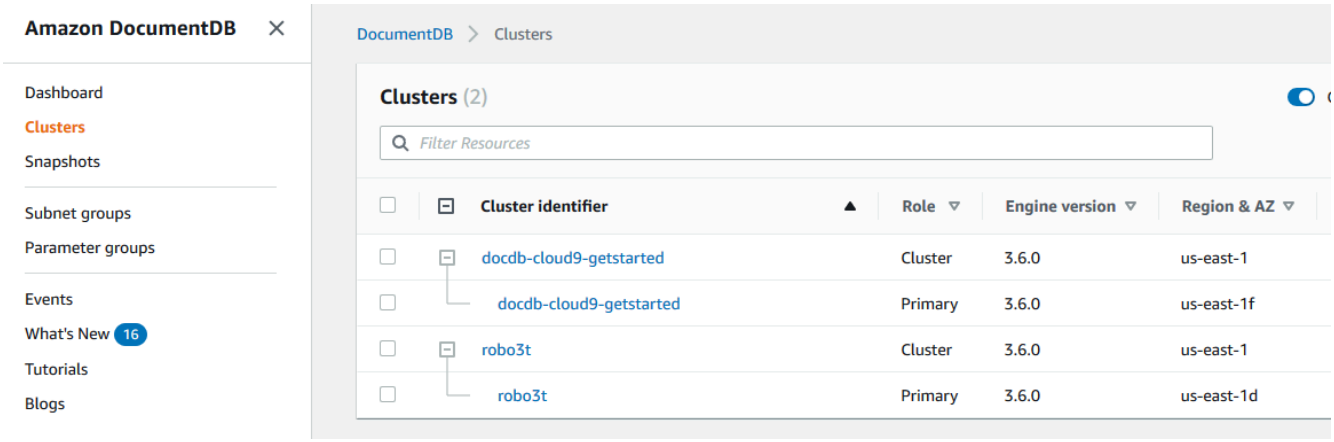
Usa Console di gestione AWS o AWS CLI per monitorare lo stato della tua istanza.

Using the Console di gestione AWS

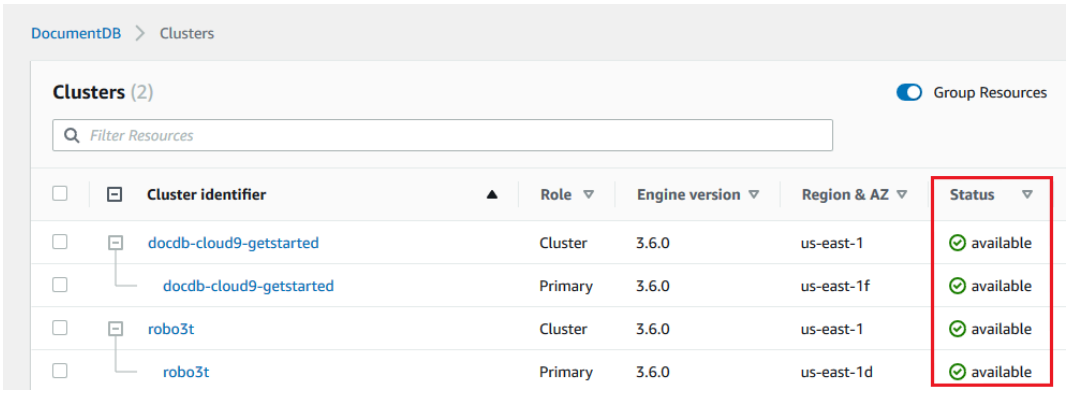
Quando si utilizza il Console di gestione AWS per determinare lo stato di un cluster, utilizzare la procedura seguente.

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.
2. Nel pannello di navigazione scegliere Cluster.

> **Note**
>
> Nota che nella casella di navigazione Clusters, la colonna Cluster identifier mostra sia i cluster che le istanze. Le istanze sono elencate sotto i cluster, in modo simile all'immagine seguente.



3. Trova il nome dell'istanza che ti interessa. Quindi, per trovare lo stato dell'istanza, leggi la riga in corrispondenza della colonna Status (Stato), come illustrato di seguito.



Using the AWS CLI

Quando si utilizza il AWS CLI per determinare lo stato di un cluster, utilizzare l'operazione. `describe-db-instances` Il codice seguente individua lo stato dell'istanza `sample-cluster-instance-01`.

Per Linux, macOS o Unix:

```
aws docdb describe-db-instances \
          --db-instance-identifier sample-cluster-instance-01  \
          --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]'
```

Per Windows:

```
aws docdb describe-db-instances ^
```

```
          --db-instance-identifier sample-cluster-instance-01  ^
          --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]'
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
[
          [
              "sample-cluster-instance-01",
              "available"
          ]
      ]
```

Valori dello stato di integrità dell'istanza

La tabella seguente elenca i possibili valori dello stato di integrità delle istanze. La colonna Instance health, situata nella tabella Clusters in Console di gestione AWS, mostra se il motore di database, il componente responsabile dell'archiviazione, della gestione e del recupero dei dati, funziona normalmente. Questa colonna indica anche se la metrica di `EngineUptime` sistema, disponibile in CloudWatch, mostra lo stato di integrità di ciascuna istanza.

Stato di integrità dell'istanza	Description
integro	Il motore di database è in esecuzione nell'istanza Amazon DocumentDB.
malsano	Il motore di database non è in esecuzione o è stato riavviato meno di un minuto fa.

Monitoraggio dello stato di integrità dell'istanza utilizzando Console di gestione AWS

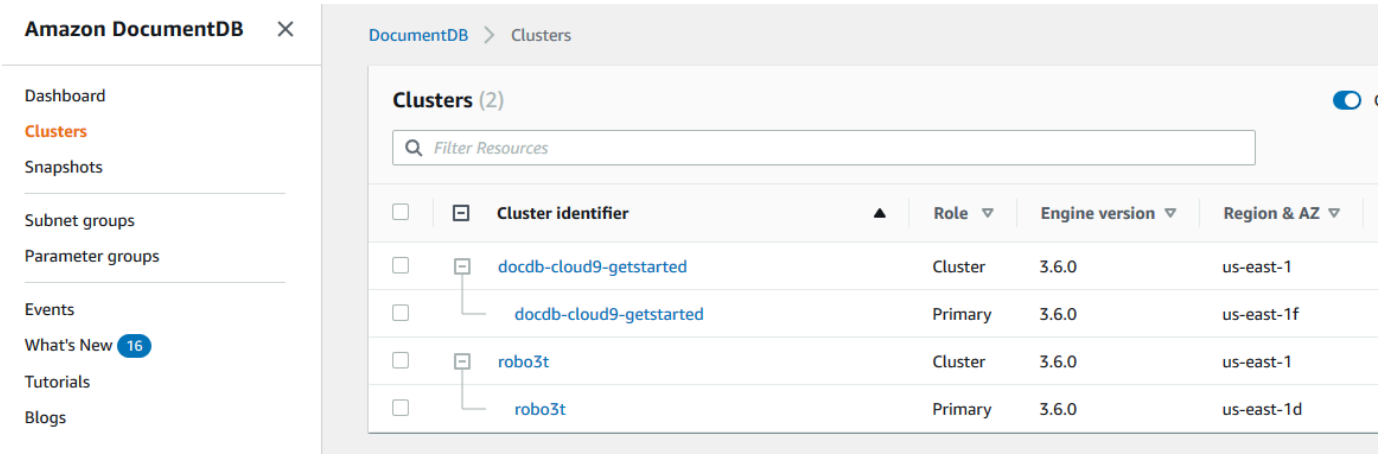
Usa il Console di gestione AWS per monitorare lo stato di integrità della tua istanza.

Durante l'utilizzo Console di gestione AWS, utilizza i seguenti passaggi per comprendere lo stato di integrità dell'istanza.

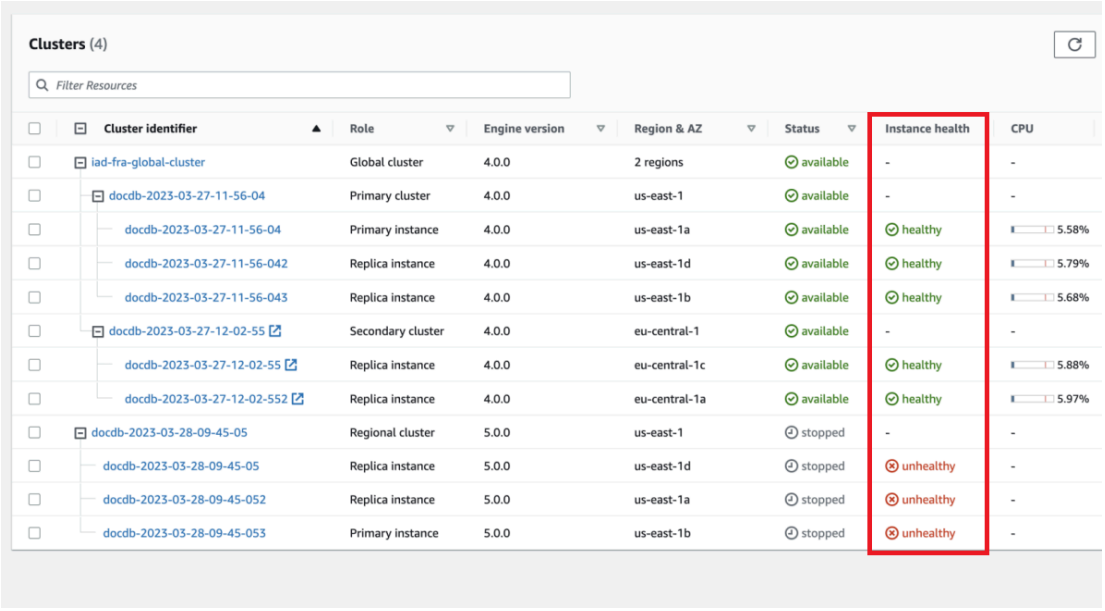
1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.

2. Nel pannello di navigazione scegliere Cluster.

> **Note**
>
> Nella casella di navigazione Clusters, la colonna Cluster identifier mostra sia i cluster che le istanze. Le istanze sono elencate sotto i cluster, in modo simile all'immagine seguente.



3. Trova il nome dell'istanza che ti interessa. Quindi, per trovare lo stato dell'istanza, leggi sull'altra riga fino alla colonna Instance health, come mostrato nell'immagine seguente:



> **Note**
>
> Il polling sullo stato di integrità delle istanze viene eseguito ogni 60 secondi e si basa sulla metrica di sistema. CloudWatch `EngineUptime` I valori nella colonna Stato dell'istanza vengono aggiornati automaticamente.

Visualizzazione dei consigli di Amazon DocumentDB

Amazon DocumentDB fornisce un elenco di consigli automatici per le risorse di database, come istanze e cluster. Questi consigli forniscono indicazioni sulle migliori pratiche analizzando le configurazioni dei cluster e delle istanze.

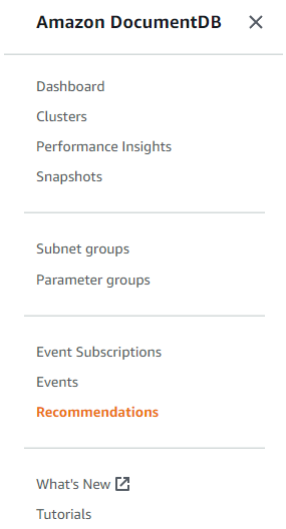
Come esempio di questi consigli, consulta quanto segue:

Tipo	Descrizione	Raccomandazione	Informazioni aggiuntive
Una istanza	Il cluster contiene solo un'istanza	Prestazioni e disponibilità: aggiungi un'altra istanza con la stessa classe di istanza in una zona di disponibilità diversa.	Disponibilità e replica elevate di Amazon DocumentDB

Amazon DocumentDB genera raccomandazioni per una risorsa quando la risorsa viene creata o modificata. Amazon DocumentDB inoltre analizza periodicamente le tue risorse e genera consigli.

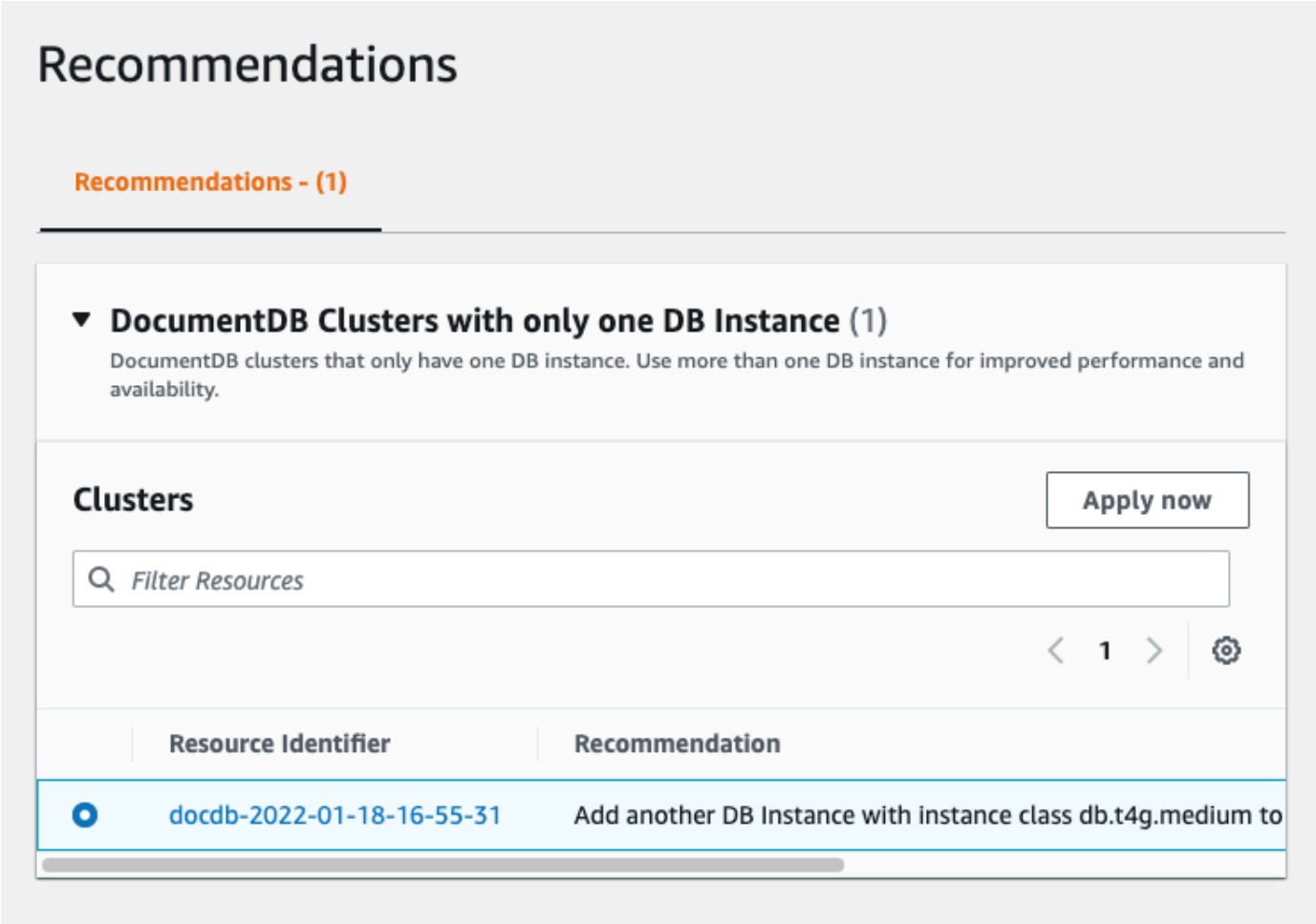
Per visualizzare e agire in base ai consigli di Amazon DocumentDB

1. Accedi e apri la Console di gestione AWS console di Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione, scegli Consigli:



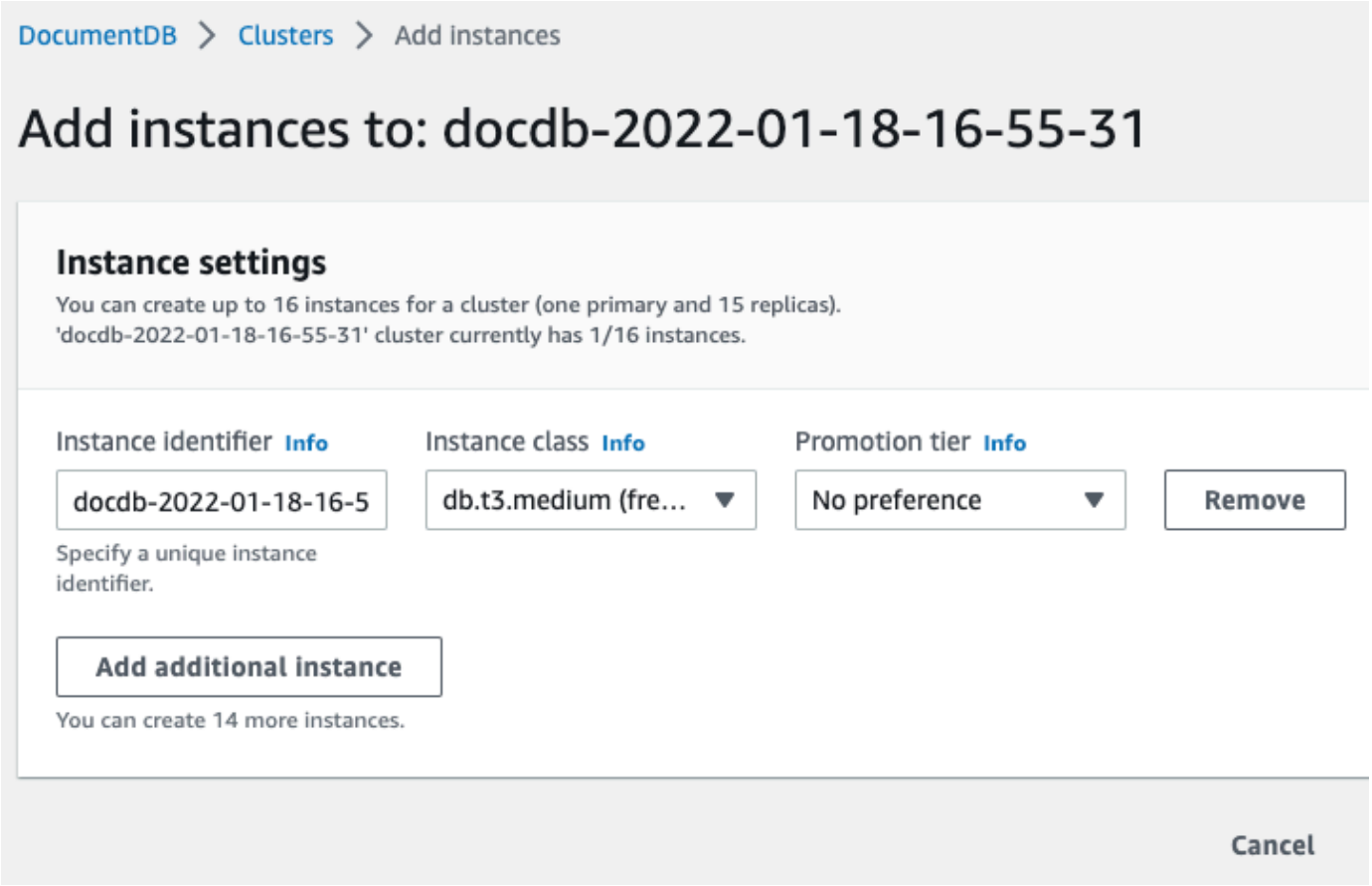
3. Nella finestra di dialogo Consigli, espandi la sezione di interesse e seleziona l'attività consigliata.

   Nell'esempio seguente, l'attività consigliata si applica a un cluster Amazon DocumentDB con una sola istanza. Si consiglia di aggiungere un'altra istanza per migliorare le prestazioni e la disponibilità.



4. Scegli Apply now (Applica ora).

   Per questo esempio, viene visualizzata la finestra di dialogo Aggiungi istanze:



5. Modifica le impostazioni della nuova istanza e scegli Crea.

Utilizzo delle sottoscrizioni agli eventi di Amazon DocumentDB

Amazon DocumentDB utilizza Amazon Simple Notification Service (Amazon SNS) per fornire notifiche quando si verifica un evento Amazon DocumentDB. Queste notifiche possono essere in qualsiasi forma supportata da Amazon SNS Regione AWS, ad esempio un'e-mail, un messaggio di testo o una chiamata a un endpoint HTTP.

Amazon DocumentDB raggruppa questi eventi in categorie a cui puoi abbonarti in modo da ricevere notifiche quando si verifica un evento in quella categoria. Puoi iscriverti a una categoria di eventi per un'istanza, un cluster, un'istantanea, un'istantanea del cluster o per un gruppo di parametri. Ad esempio, se ti iscrivi alla categoria Backup per una determinata istanza, ricevi una notifica ogni volta che si verifica un evento relativo al backup che influisce sull'istanza. Riceverai anche una notifica quando la sottoscrizione di un evento cambia.

Gli eventi si verificano sia a livello di cluster che a livello di istanza, quindi puoi riceverli se ti iscrivi a un cluster o a un'istanza.

Le sottoscrizioni agli eventi vengono inviate agli indirizzi forniti al momento della creazione dell'abbonamento. Potresti voler creare diversi abbonamenti, ad esempio un abbonamento che riceve tutte le notifiche degli eventi e un altro abbonamento che include solo eventi critici per le istanze di produzione. Puoi disattivare facilmente la notifica senza eliminare una sottoscrizione. A tale scopo, imposta il pulsante di opzione Enabled su No nella console Amazon DocumentDB.

> **Important**
>
> Amazon DocumentDB non garantisce l'ordine degli eventi inviati in un flusso di eventi. Tale ordine è soggetto a modifiche.

Amazon DocumentDB utilizza l'Amazon Resource Name (ARN) di un argomento Amazon SNS per identificare ogni abbonamento. La console Amazon DocumentDB crea automaticamente l'ARN al momento della creazione dell'abbonamento.

La fatturazione per gli abbonamenti agli eventi Amazon DocumentDB avviene tramite Amazon SNS. L'uso della notifica degli eventi è soggetta alle tariffe di Amazon SNS. Per ulteriori informazioni, consulta Prezzi di Amazon Simple Notification Service. Oltre ai costi di Amazon SNS, Amazon DocumentDB non fattura gli abbonamenti agli eventi.

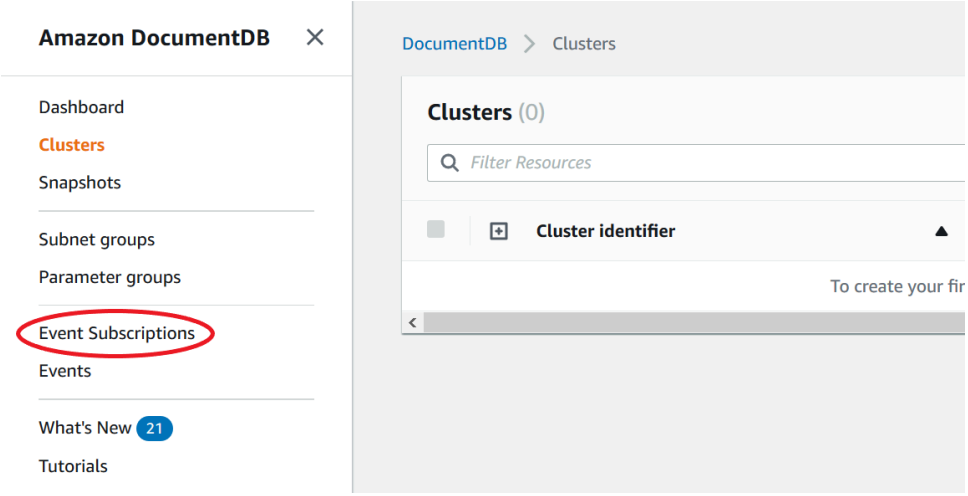
Argomenti

- Iscrizione agli eventi di Amazon DocumentDB
- Gestione degli abbonamenti per le notifiche di eventi di Amazon DocumentDB
- Categorie e messaggi di eventi di Amazon DocumentDB

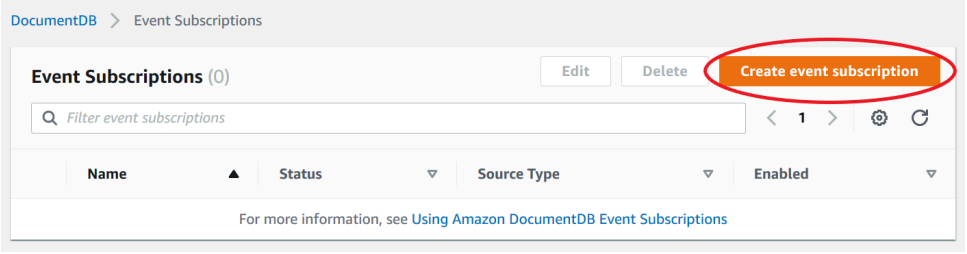
Iscrizione agli eventi di Amazon DocumentDB

Puoi utilizzare la console Amazon DocumentDB per sottoscrivere sottoscrizioni a eventi, come segue:

1. Accedi all'indirizzo. Console di gestione AWS https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione selezionare Event subscriptions (Sottoscrizioni di eventi).

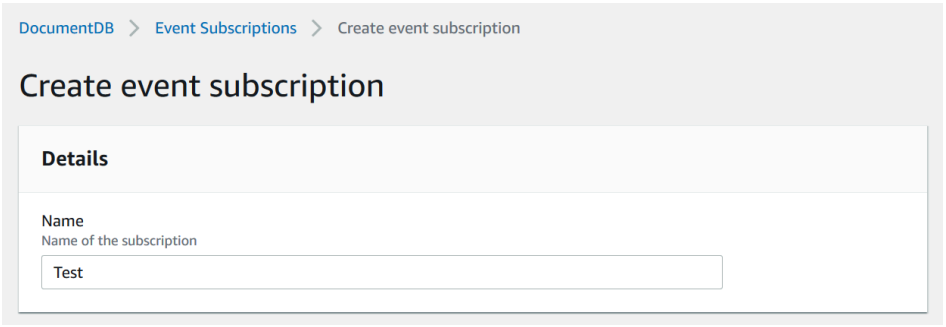


3. Nel riquadro Event subscriptions (Sottoscrizioni di eventi) scegliere Create event subscription (Crea sottoscrizione di eventi).

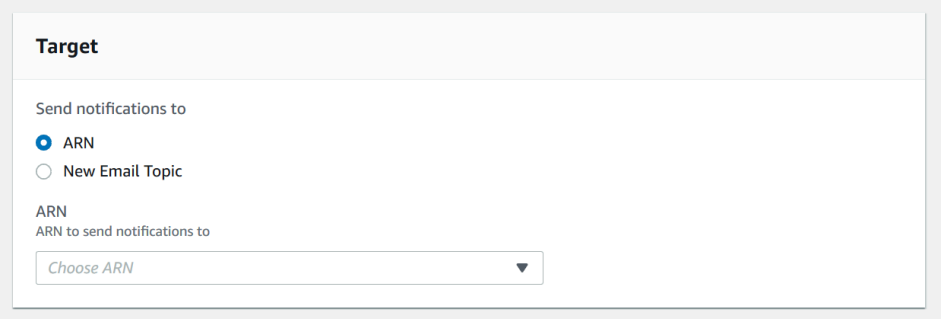


4. Nella finestra di dialogo Create event subscription (Crea sottoscrizione di eventi), seguire questa procedura:

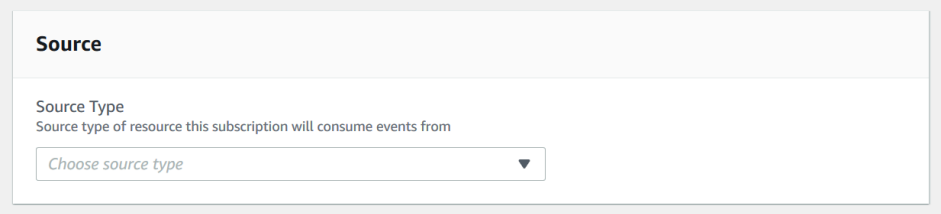
   - Per Name (Nome), immettere un nome per la sottoscrizione alle notifiche eventi.



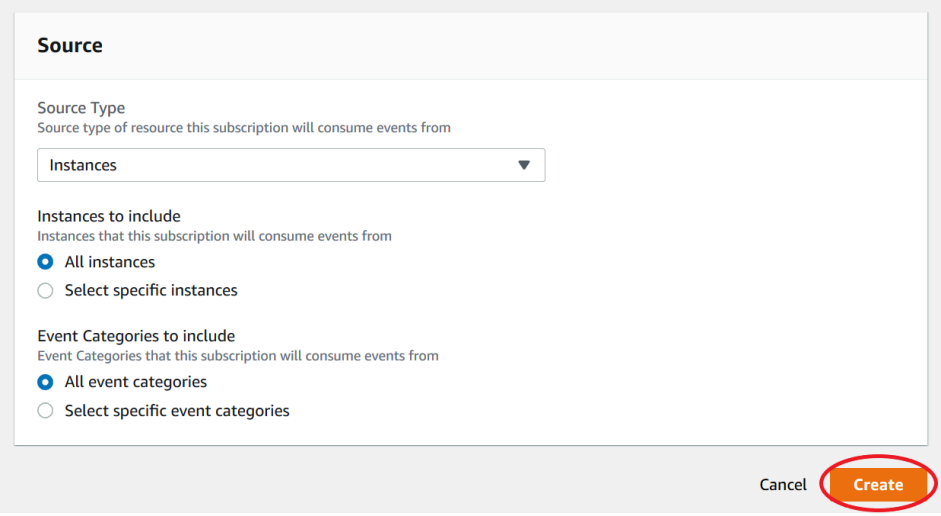
   - Per Target, scegli a chi inviare le notifiche. Puoi scegliere un ARN esistente o scegliere Nuovo argomento e-mail per inserire il nome di un argomento e un elenco di destinatari.



- Per Sorgente, scegli un tipo di fonte. A seconda del tipo di origine selezionato, scegliere le categorie di eventi e le origini da cui ricevere le notifiche eventi.



- Scegli Create (Crea).

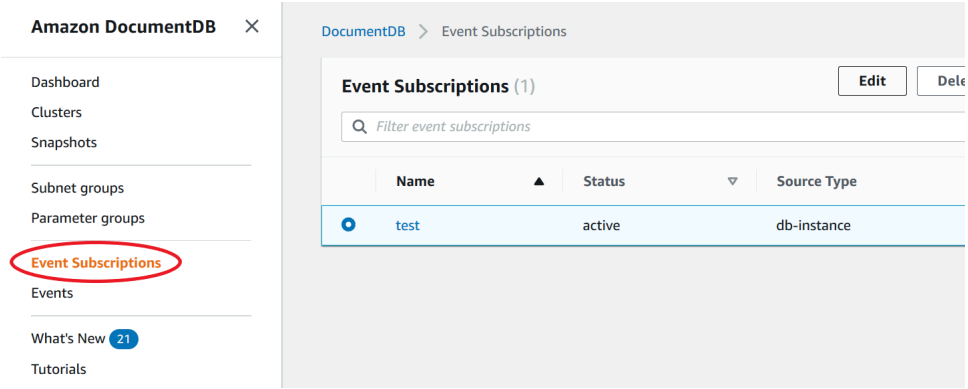


Gestione degli abbonamenti per le notifiche di eventi di Amazon DocumentDB

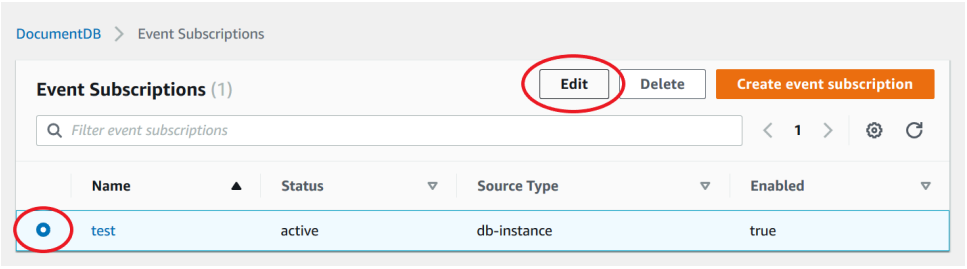
Se scegli Abbonamenti a eventi nel pannello di navigazione della console Amazon DocumentDB, puoi visualizzare le categorie di abbonamento e un elenco degli abbonamenti correnti. Puoi anche modificare o eliminare una sottoscrizione specifica.

Per modificare gli attuali abbonamenti di notifica degli eventi di Amazon DocumentDB

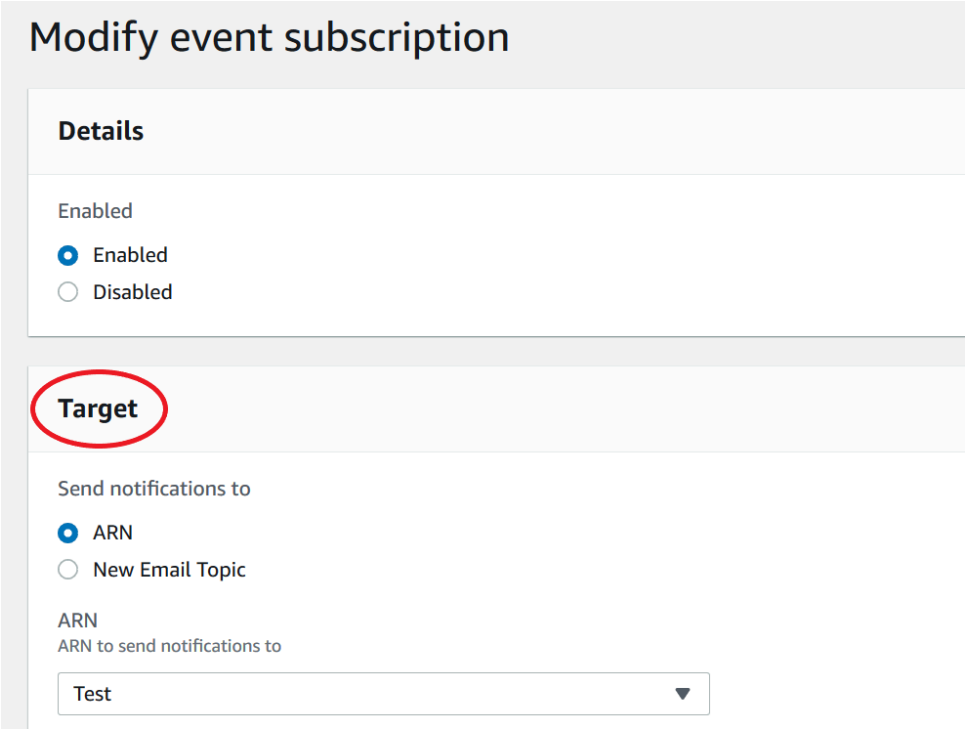
1. Accedi all'indirizzo. Console di gestione AWS https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione selezionare Event subscriptions (Sottoscrizioni di eventi). Il riquadro Event subscriptions (Sottoscrizioni di eventi) mostra tutte le sottoscrizioni delle notifiche degli eventi.



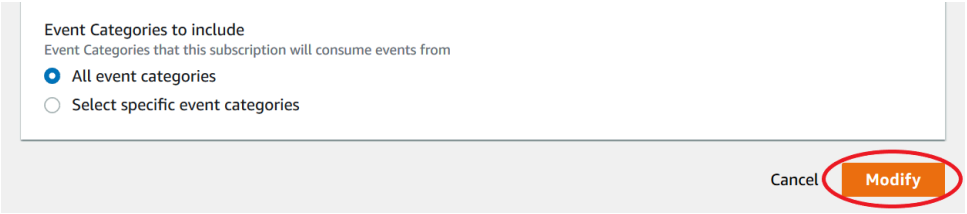
3. Nel riquadro Event subscriptions (Sottoscrizioni di eventi) scegliere la sottoscrizione da modificare e selezionare Edit (Modifica).



4. Apportare le modifiche alla sottoscrizione nella sezione Target (Destinazione) o Source (Origine). Puoi aggiungere o rimuovere gli identificatori di origine selezionandoli o deselezionandoli nella sezione Origine.



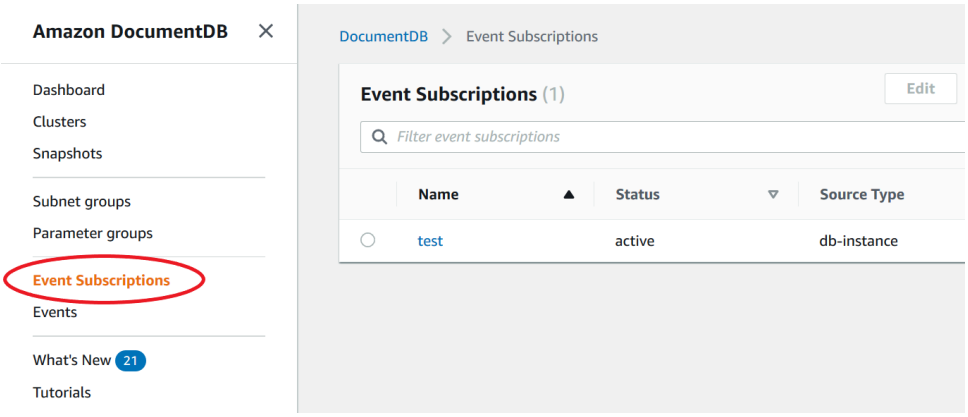
5. Scegli Modifica. La console Amazon DocumentDB indica che l'abbonamento è in fase di modifica.



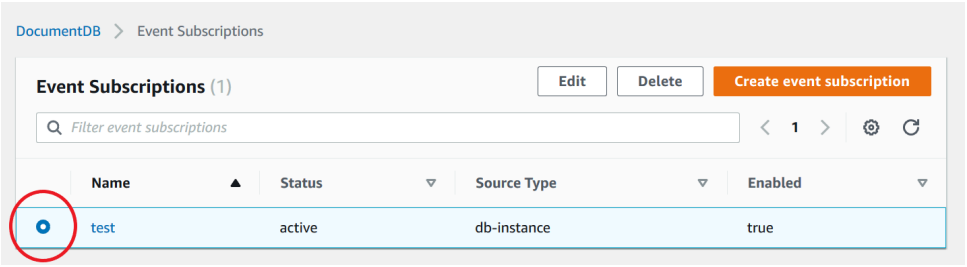
Eliminazione di una sottoscrizione di notifica di eventi Amazon DocumentDB

Puoi eliminare un abbonamento quando questo non è più necessario. Tutti gli abbonati all'argomento non riceveranno più le notifiche di eventi specificate dall'abbonamento.

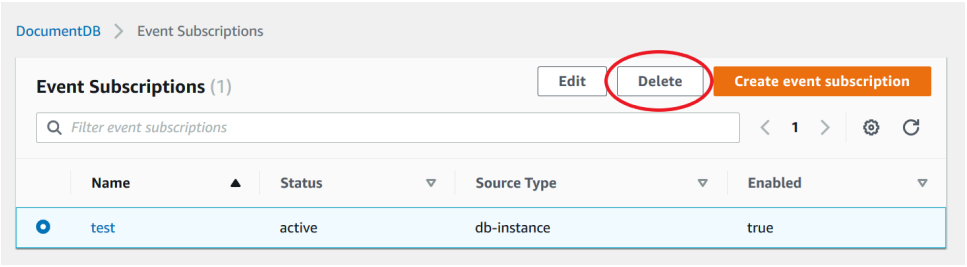
1. Accedi all'indirizzo. Console di gestione AWS https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione selezionare Event subscriptions (Sottoscrizioni di eventi).



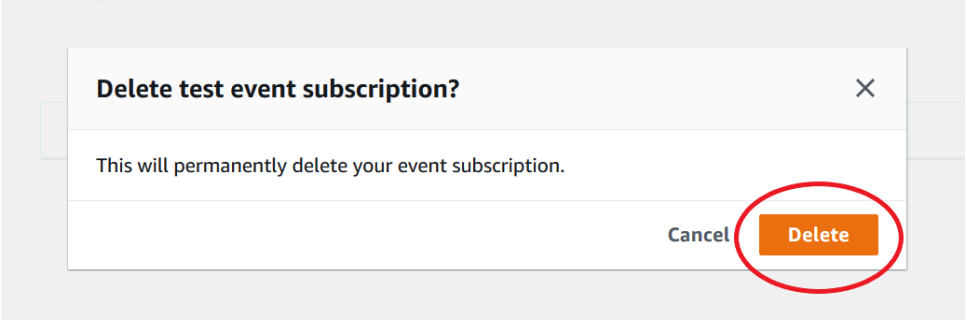
3. Nel riquadro Sottoscrizione a eventi scegliere la sottoscrizione che si vuole eliminare.



4. Scegli Elimina.



5. Apparirà una finestra pop-up che ti chiederà se desideri eliminare definitivamente questa notifica. Scegli Elimina.



Categorie e messaggi di eventi di Amazon DocumentDB

Amazon DocumentDB genera un numero significativo di eventi in categorie a cui è possibile abbonarsi utilizzando la console. Ogni categoria si applica a un tipo di sorgente, che può essere un'istanza, un cluster, un'istantanea o un gruppo di parametri.

> **Note**
>
> Amazon DocumentDB utilizza le definizioni e gli ID di eventi Amazon RDS esistenti.

Eventi Amazon DocumentDB provenienti da istanze

Categoria	Description
availability	L'istanza è stata riavviata.
disponibilità	La chiusura dell'istanza.
modifica della configurazione	Applicazione della modifica a una classe di istanza.
modifica della configurazione	È stata completata l'applicazione della modifica a una classe di istanza.
modifica della configurazione	Reimposta le credenziali primarie.
creazione	Istanza creata.
eliminazione	Istanza eliminata
errore	L'istanza non è riuscita a causa di una configurazione incompatibile o di un problema di archiviazione sottostante. Inizia un ripristino temporizzato per l'istanza.
notification	Istanza interrotta.
notification	Istanza avviata.

Categoria	Description
notification	L'istanza viene avviata a causa del superamento del tempo massimo consentito per l'interruzione.
recupero	Il ripristino dell'istanza è iniziato. La durata del recupero varia in funzione della quantità di dati da recuperare.
recupero	Il ripristino dell'istanza è completo.
applicazione di patch di sicurezza	L'aggiornamento del sistema operativo è disponibile per la tua istanza. Per informazioni sull'applicazione degli aggiornamenti, consulta Manutenzione di Amazon DocumentDB.

Eventi Amazon DocumentDB provenienti da un cluster

Categoria	Description
creazione	Cluster creato
eliminazione	Cluster eliminato.
failover	Promuovere nuovamente le primarie precedenti.
failover	Failover sull'istanza completato.
failover	Failover avviato sull'istanza DB: %s
failover	È stato avviato il failover dello stesso tipo AZ sull'istanza DB: %s
failover	Avviato il failover cross AZ verso l'istanza DB: %s

Categoria	Description
manutenzione	Il cluster è stato patchato.
manutenzione	Il cluster di database è in uno stato che non può essere aggiornato: %s
notification	Il cluster si è fermato.
notification	Il cluster è iniziato.
notification	L'arresto del cluster non è riuscito.
notification	Il cluster viene avviato a causa del superamento del tempo massimo consentito per l'interruzione.
notification	Cluster rinominato da %s a %s.

Eventi Amazon DocumentDB provenienti da uno snapshot del cluster

La tabella seguente mostra la categoria di eventi e un elenco di eventi quando uno snapshot del cluster Amazon DocumentDB è il tipo di origine.

Categoria	Description
backup	Creazione di uno snapshot del cluster manuale in corso.
backup	Snapshot del cluster manuale creato.
backup	Creazione snapshot cluster automatizzato.
backup	Snapshot di cluster automatizzato creato.

Eventi Amazon DocumentDB originati dal gruppo di parametri

Nella tabella seguente sono indicati la categoria di evento e un elenco di eventi quando il tipo di origine è un gruppo di parametri.

Categoria	Description
modifica della configurazione	Parametro aggiornato da %s a %s con il metodo di applicazione %s

Monitoraggio di Amazon DocumentDB con CloudWatch

Amazon DocumentDB (con compatibilità con MongoDB) si integra con Amazon in CloudWatch modo da poter raccogliere e analizzare le metriche operative per i tuoi cluster. Puoi monitorare queste metriche utilizzando la CloudWatch console, la console Amazon DocumentDB, la () o l'API. AWS Command Line Interface AWS CLI CloudWatch

CloudWatch consente inoltre di impostare allarmi in modo da poter ricevere una notifica se un valore della metrica supera una soglia specificata. Puoi persino configurare Amazon CloudWatch Events per intraprendere azioni correttive in caso di violazione. Per ulteriori informazioni sull'utilizzo CloudWatch e sugli allarmi, consulta la documentazione di Amazon CloudWatch .

Argomenti

- Metriche di Amazon DocumentDB
- Visualizzazione CloudWatch dei dati
- Dimensioni di Amazon DocumentDB
- Monitoraggio delle metriche opcounter
- Monitoraggio delle connessioni al database

Metriche di Amazon DocumentDB

Per monitorare lo stato e le prestazioni del cluster e delle istanze Amazon DocumentDB, puoi visualizzare le seguenti metriche nella console Amazon DocumentDB.

> **Note**
>
> Le metriche nelle tabelle seguenti si applicano sia ai cluster basati su istanze che a quelli elastici.

Argomenti

- Parametri di utilizzo delle risorse
- Metriche di latenza
- NVMe-backed metriche dell'istanza
- Metriche operative
- Command-level metriche
- Metriche di throughput
- Parametri del sistema
- Metriche delle istanze T3

Parametri di utilizzo delle risorse

Metrica	Description
`BackupRetentionPeriodStorageUsed`	La quantità totale di storage di backup in byte utilizzata per supportare la funzionalità di ripristino point-in-time all'interno della finestra di conservazione di Amazon DocumentDB. Incluso nel totale riportato dal parametro `TotalBackupStorageBilled` . Calcolato separatamente per ogni cluster Amazon DocumentDB.
`ChangeStreamLogSize`	La quantità di storage utilizzata dal cluster per archiviare il log del flusso di modifiche in megabyte. Questo valore è un sottoinsieme dello storage totale per il cluster (`VolumeBytesUsed`) e influisce sul costo del cluster. Per informazioni sui prezzi

Metrica	Description
	dello storage, consulta la pagina del prodotto Amazon DocumentDB. La dimensione del log del flusso di modifiche è una funzione della quantità di modifiche applicata nel cluster e della durata di conservazione prolungata del flusso di modifiche. Per ulteriori informazioni sui flussi di modifiche, consulta Utilizzo dei flussi di modifiche con Amazon DocumentDB.
`CPUUtilization`	La percentuale di CPU utilizzata da un'istanza.
`DatabaseConnections`	Il numero di connessioni (attive e inattive) aperte su un'istanza eseguita con una frequenza di 1 minuto.
`DatabaseConnectionsMax`	Il numero massimo di connessioni al database aperte (attive e inattive) su un'istanza in un periodo di 1 minuto.
`DatabaseConnectionsLimit`	Il numero massimo di connessioni simultanee al database (attive e inattive) consentite su un'istanza in un dato momento.

Metrica	Description
`DatabaseConnectionsActiveMax`	Il numero massimo di connessioni al database attive su un'istanza in un periodo di 1 minuto. Le connessioni attive sono quelle attualmente in esecuzione di un'operazione, a differenza delle connessioni inattive. Per le quote di connessione attive per tipo di istanza, vedere. Quote delle istanze
`DatabaseCursors`	Il numero di cursori aperti su un'istanza acquisita con una frequenza di 1 minuto.
`DatabaseCursorsMax`	Il numero massimo di cursori aperti su un'istanza in un periodo di 1 minuto.
`DatabaseCursorsLimit`	Il numero massimo di cursori consentiti su un'istanza in un dato momento.
`DatabaseCursorsTimedOut`	Il numero di cursori scaduti in un periodo di 1 minuto.
`FreeableMemory`	La quantità di memoria RAM disponibile, in byte.

Metrica	Description
FreeLocalStorage	Questo parametro indica la quantità di storage disponibi le in ogni istanza per le tabelle e i log temporanei. Questo valore dipende dalla classe di istanza. Puoi aumentare la quantità di storage gratuito per un'istanza scegliendo una classe di istanza più ampia per l'istanza. (Questo non si applica a DocumentDB Serverless.)
LowMemThrottleQueueDepth	La profondità della coda per le richieste limitate a causa della scarsa memoria disponibile utilizzata con una frequenza di 1 minuto.
LowMemThrottleMaxQueueDepth	La profondità massima della coda per le richieste limitate a causa della scarsa memoria disponibile in un periodo di 1 minuto.
LowMemNumOperationsThrottled	Il numero di richieste limitate a causa della scarsa memoria disponibile in un periodo di 1 minuto.

Metrica	Description
SnapshotStorageUsed	La quantità totale di storage di backup, in byte, consumata da tutte le istantanee per un determinato cluster Amazon DocumentDB al di fuori della finestra di conservazione dei backup. Incluso nel totale riportato dal parametro TotalBackupStorageBilled . Calcolato separatamente per ogni cluster Amazon DocumentDB.
SwapUsage	La quantità di spazio di swapping utilizzato sull'istanza.
TotalBackupStorageBilled	La quantità totale di storage di backup in byte per la quale ti viene fatturato un determinato cluster Amazon DocumentDB. Include lo storage di backup misurato dai parametri BackupRetentionPeriodStorageUsed e SnapshotStorageUsed . Calcolato separatamente per ogni cluster Amazon DocumentDB.
TransactionsOpen	Il numero di transazioni aperte su un'istanza eseguita con una frequenza di 1 minuto.

Metrica	Description	
`TransactionsOpenMax`	Il numero massimo di transazioni aperte su un'istanza in un periodo di 1 minuto.	
`TransactionsOpenLimit`	Il numero massimo di transazioni simultanee consentite su un'istanza in un dato momento.	
`VolumeBytesUsed`	La quantità di storage in byte utilizzata dal cluster. Questo valore influisce sul costo del cluster. Per informazioni sui prezzi, consulta la pagina del prodotto Amazon DocumentDB.	

Metriche di latenza

Metrica	Description	
`DBClusterReplicaLagMaximum`	La quantità massima di ritardo, in millisecondi, tra l'istanza primaria e ogni istanza Amazon DocumentDB nel cluster.	
`DBClusterReplicaLagMinimum`	Il ritardo minimo, in millisecondi, tra l'istanza primaria e ogni istanza di replica nel cluster.	
`DBInstanceReplicaLag`	Il ritardo, in millisecondi, durante la replica degli aggiornamenti dall'istanza	

Metrica	Description
	primaria a un'istanza di replica.
`ReadLatency`	Il tempo medio impiegato per operazione su disco. I/O
`WriteLatency`	Il tempo medio, in millisecondi, impiegato per operazione su disco. I/O

NVMe-backed metriche dell'istanza

Metrica	Description
`NVMeStorageCacheHitRatio`	La percentuale di richieste servite dalla cache a più livelli.
`FreeNVMeStorage`	La quantità di archiviazione NVMe effimera disponibile.
`ReadIOPSNVMeStorage`	Il numero medio di I/O operazioni di lettura del disco sullo storage Ephemeral NVMe.
`ReadLatencyNVMeStorage`	Il tempo medio impiegato per I/O operazione di lettura del disco per lo storage Ephemeral NVMe.
`ReadThroughputNVMeStorage`	Il numero medio di byte letti dal disco al secondo per l'archiviazione NVMe effimera.
`WriteIOPSNVMeStorage`	Il numero medio di I/O operazioni di scrittura su

Metrica	Description
	disco sullo storage Ephemeral NVMe.
`WriteLatencyNVMeStorage`	Il tempo medio impiegato per I/O operazione di scrittura su disco per lo storage Ephemeral NVMe.
`WriteThroughputNVMeStorage`	Il numero medio di byte scritti sul disco al secondo per l'archiviazione NVMe effimera.

Metriche operative

Metrica	Description
`DocumentsDeleted`	Il numero di documenti eliminati in un periodo di 1 minuto.
`DocumentsInserted`	Il numero di documenti inseriti in un periodo di 1 minuto.
`DocumentsReturned`	Il numero di documenti restituiti in un periodo di 1 minuto.
`DocumentsUpdated`	Il numero di documenti aggiornati in un periodo di 1 minuto.
`OpcountersCommand`	Il numero di comandi emessi in un periodo di 1 minuto.
`OpcountersDelete`	Il numero di operazioni di eliminazione eseguite in un periodo di 1 minuto.

Metrica	Description
`OpcountersGetmore`	Il numero di getmore emessi in un periodo di 1 minuto.
`OpcountersInsert`	Il numero di operazioni di inserimento effettuate in un periodo di 1 minuto.
`OpcountersQuery`	Il numero di interrogazioni effettuate in un periodo di 1 minuto.
`OpcountersUpdate`	Il numero di operazioni di aggiornamento eseguite in un periodo di 1 minuto.
`TransactionsStarted`	Il numero di transazioni avviate su un'istanza in un periodo di 1 minuto.
`TransactionsCommitted`	Il numero di transazioni effettuate su un'istanza in un periodo di 1 minuto.
`TransactionsAborted`	Il numero di transazioni interrotte su un'istanza in un periodo di 1 minuto.
`TTLDeletedDocuments`	Il numero di documenti eliminati da un TTLMonitor in un periodo di 1 minuto.

Command-level metriche

Amazon DocumentDB pubblica le metriche di latenza e concorrenza per comando su. CloudWatch Queste metriche ti aiutano a identificare le operazioni che contribuiscono maggiormente al tuo carico di lavoro e a rilevare problemi di latenza finale o pressioni sulla concorrenza.

Ogni comando nella tabella seguente emette tre metriche in un periodo di 1 minuto:

- `AvgDuration`— Il tempo medio di esecuzione del comando, in microsecondi.
- `P100Duration`— Il tempo massimo di esecuzione del comando (100° percentile), in microsecondi.
- `MaxConcurrent`— Il numero massimo di esecuzioni simultanee del comando.

I nomi delle metriche seguono lo schema. `Command.`*`commandName`*`.`*`metricName`* Ad esempio `Command.find.AvgDuration` o `Command.aggregate.MaxConcurrent`.

Metrica	Description
`Command.find`	Leggi le query utilizzando find.
`Command.insert`	Operazioni di inserimento di documenti.
`Command.findAndModify`	Operazioni di ricerca e modifica atomiche.
`Command.update`	Operazioni di aggiornamento dei documenti.
`Command.delete`	Operazioni di eliminazione dei documenti.
`Command.aggregate`	Esecuzioni della pipeline di aggregazione.
`Command.count`	Operazioni di conteggio.
`Command.distinct`	Interrogazioni con valori distinti.
`Command.getMore`	Operazioni di recupero in batch con cursore.
`Command.abortTransaction`	Operazioni di interruzione della transazione.
`Command.commitTransaction`	Operazioni di commit delle transazioni.

Metriche di throughput

Metrica	Description	
`NetworkReceiveThroughput`	La quantità di throughput della rete, in byte al secondo, ricevuta dai clienti per ogni istanza nel cluster. Questo throughput non include il traffico di rete tra le istanze nel cluster e il volume cluster.	
`NetworkThroughput`	La quantità di throughput di rete, in byte al secondo, ricevuta e trasmessa ai client da ciascuna istanza del cluster Amazon DocumentDB. Questo throughput non include il traffico di rete tra le istanze nel cluster e il volume cluster.	
`NetworkTransmitThroughput`	La quantità di throughput della rete, in byte al secondo, inviata ai clienti per ogni istanza nel cluster. Questo throughput non include il traffico di rete tra le istanze nel cluster e il volume cluster.	
`ReadIOPS`	Il numero medio di I/O operazioni di lettura del disco al secondo. I report di Amazon DocumentDB leggono e scrivono gli IOPS separatam ente e a intervalli di 1 minuto.	

Metrica	Description
`ReadThroughput`	Il numero medio di byte letti dal disco al secondo.
`StorageNetworkReceiveThroughput`	La quantità di throughput di rete, in byte al secondo, ricevuta dal volume di storage del cluster Amazon DocumentDB da ciascuna istanza del cluster.
`StorageNetworkTransmitThroughput`	La quantità di throughput di rete, in byte al secondo, inviata al volume di storage del cluster Amazon DocumentDB da ciascuna istanza del cluster.
`StorageNetworkThroughput`	La quantità di throughput di rete, in byte al secondo, ricevuta e inviata al volume di storage del cluster Amazon DocumentDB da ciascuna istanza del cluster Amazon DocumentDB.

Metrica	Description
VolumeReadIOPs	Il numero medio di I/O operazioni di lettura fatturate da un volume del cluster, riportate a intervalli di 5 minuti. Le operazioni di lettura fatturate sono calcolate a livello del volume del cluster, aggregate da tutte le istanze nel cluster e quindi indicate a intervalli di 5 minuti. Il valore viene calcolato prendendo il valore del parametro per le operazioni di lettura in un periodo di 5 minuti. Puoi determinare la quantità delle operazioni di lettura fatturate al secondo prendendo il valore del parametro per le operazioni di lettura fatturate e dividendolo per 300 secondi. Ad esempio, se VolumeReadIOPs restituisce 13.686, le operazioni di lettura fatturate al secondo sono 45 (13.686/300 = 45,62). Le operazioni di lettura fatturate si accumulano per le query che richiedono pagine del database che non si trovano nella cache del buffer e, per questo, devono essere caricate dallo storage. Potresti vedere dei picchi nelle

Metrica	Description	
	operazioni di lettura fatturate poiché i risultati della query vengono letti dallo storage e, in seguito, caricati nella cache del buffer.	

Metrica	Description	
`VolumeWriteIOPs`	Il numero medio di I/O operazioni di scrittura fatturate da un volume del cluster, riportate a intervalli di 5 minuti. Le operazioni di scrittura fatturate sono calcolate a livello del volume del cluster, aggregate da tutte le istanze nel cluster e quindi indicate a intervalli di 5 minuti. Il valore è calcolato prendendo il valore del parametro delle operazioni di scrittura per un periodo che supera i 5 minuti. Puoi determinare la quantità delle operazioni di scrittura fatturate al secondo prendendo il valore del parametro delle operazioni di scrittura fatturate e dividendo per 300 secondi. Ad esempio, se `VolumeWriteIOPs` restituisce 13.686, le operazioni di scrittura fatturate al secondo sono 45 (13.686/300 = 45,62). Tieni presente che `VolumeReadIOPs` le `VolumeWriteIOPs` metriche sono calcolate dal livello di archiviazione DocumentDB e includono gli iOS eseguiti dalle istanze primarie e di replica. I dati	

Metrica	Description	
	vengono aggregati ogni 20-30 minuti e quindi riportati a intervalli di 5 minuti, emettendo così lo stesso punto dati per la metrica nel periodo di tempo. Se stai cercando una metrica da correlare alle tue operazioni di inserimento su un intervall o di 1 minuto, puoi utilizzare la metrica WriteIOPS a livello di istanza. La metrica è disponibi le nella scheda di monitorag gio dell'istanza primaria di Amazon DocumentDB.	
`WriteIOPS`	Il numero medio di I/O operazioni di scrittura su disco al secondo. Se utilizzati a livello di cluster, `WriteIOPs` vengono valutati in tutte le istanze del cluster. Gli IOPS di lettura e scrittura vengono riportati separatamente, a intervalli di 1 minuto.	
`WriteThroughput`	Il numero medio di byte scritti sul disco al secondo.	

Parametri del sistema

Metrica	Description	
`AvailableMVCCIds`	Un contatore che mostra il numero di operazioni di	

Metrica	Description	
	scrittura rimanenti disponibili prima di raggiungere lo zero. Quando questo contatore raggiunge lo zero, il cluster entrerà in modalità di sola lettura fino a quando gli ID non verranno recuperati e riciclati . Il contatore diminuisce a ogni operazione di scrittura e aumenta man mano che la Garbage Collection ricicla i vecchi ID MVCC.	
`BufferCacheHitRatio`	La percentuale di richieste gestite dalla cache del buffer.	
`DiskQueueDepth`	Il numero di I/O operazioni in attesa di essere scritte o lette dal disco.	
`EngineUptime`	Il periodo di esecuzione dell'istanza, in secondi.	
`IndexBufferCacheHitRatio`	La percentuale di richieste di indice servite dalla buffer cache. Potresti vedere un picco superiore al 100 percento per la metrica subito dopo aver eliminato un indice, una raccolta o un database. Questo verrà corretto automaticamente dopo 60 secondi. Questa limitazione verrà corretta in un futuro aggiornamento della patch.	

Metrica	Description
`LongestActiveGCRuntime`	Durata in secondi del processo di raccolta dei rifiuti attivo più lungo. Si aggiorna ogni minuto e tiene traccia solo delle operazioni attive, esclusi i processi che vengono completati entro un minuto.

Metriche delle istanze T3

Metrica	Description
`CPUCreditUsage`	Il numero di crediti CPU spesi durante il periodo di misurazione.
`CPUCreditBalance`	Il numero di crediti CPU accumulati da un'istanza. Questo saldo è esaurito quando la CPU ottimizza le prestazioni e i crediti CPU vengono spesi più rapidamente di quanto guadagnati.
`CPUSurplusCreditBalance`	Il numero di crediti CPU in eccesso spesi per sostenere le prestazioni della CPU quando il CPUCreditBalance valore è zero.
`CPUSurplusCreditsCharged`	Il numero di crediti CPU in eccesso che supera il numero massimo di crediti CPU che possono essere guadagnati in un periodo di

Metrica	Description	
	24 ore e comporta quindi un costo aggiuntivo. Per ulteriori informazioni, consulta Monitoraggio dei crediti CPU.	

Visualizzazione CloudWatch dei dati

Puoi visualizzare CloudWatch i dati Amazon utilizzando la CloudWatch console, la console Amazon DocumentDB AWS Command Line Interface (AWS CLI) o l' CloudWatch API.

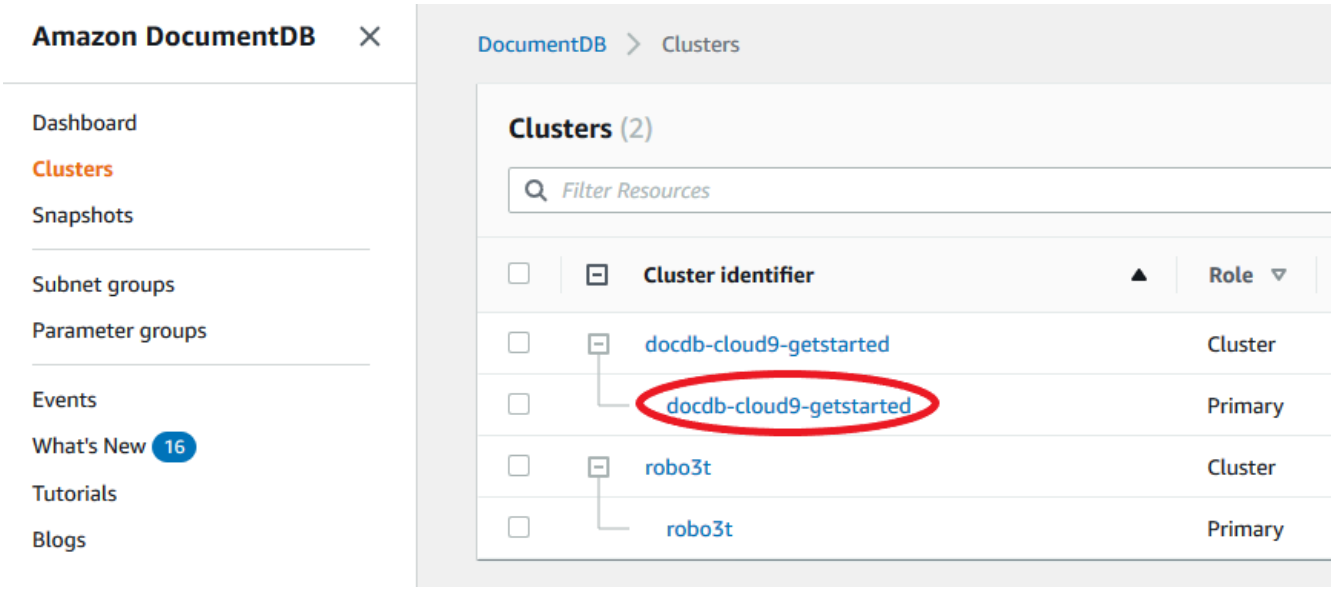
Using the Console di gestione AWS

Per visualizzare le CloudWatch metriche utilizzando la console di gestione Amazon DocumentDB, completa i seguenti passaggi.

1. Accedi e apri la console Console di gestione AWS di Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.

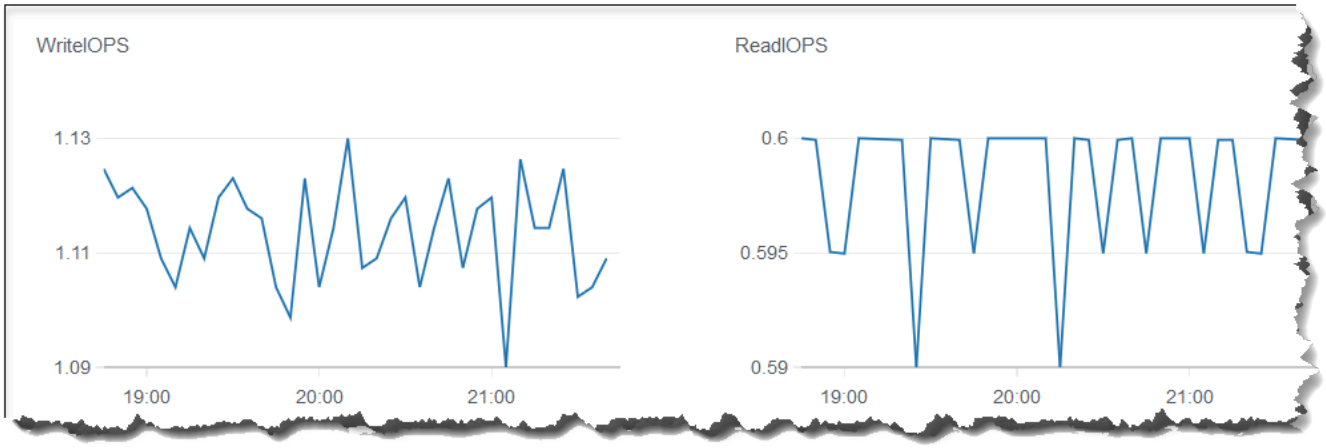
   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

3. Nella casella di navigazione Clusters, vedrai la colonna Cluster Identifier. Le tue istanze sono elencate in cluster, in modo simile alla schermata seguente.



4. Dall'elenco delle istanze, scegli il nome dell'istanza per la quale desideri le metriche.
5. Nella pagina di riepilogo dell'istanza risultante, scegli la scheda Monitoraggio per visualizzare le rappresentazioni grafiche delle metriche della tua istanza Amazon DocumentDB. Poiché è necessario generare un grafico per ogni metrica, la compilazione dei grafici potrebbe richiedere alcuni minuti. CloudWatch

   L'immagine seguente mostra le rappresentazioni grafiche di due CloudWatch metriche nella console Amazon DocumentDB e. `WriteIOPS` `ReadIOPS`



Using the CloudWatch Management Console

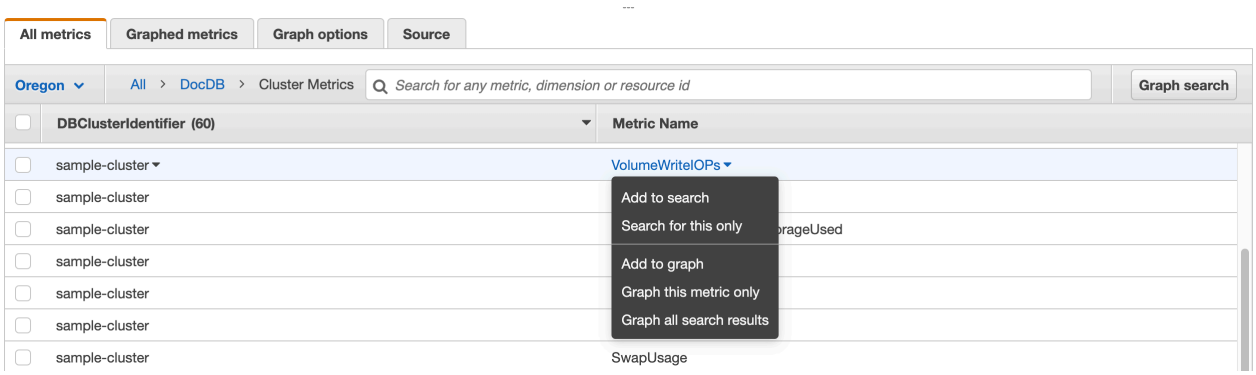
Per visualizzare le CloudWatch metriche utilizzando la console di gestione, completa i seguenti passaggi CloudWatch .

1. Accedi e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/cloudwatch
2. Nel riquadro di navigazione, seleziona Parametri. Quindi, dall'elenco dei nomi dei servizi, scegli DocDB.
3. Scegli una dimensione metrica (ad esempio, Cluster Metrics).
4. La scheda Tutte le metriche mostra tutte le metriche per quella dimensione in DocDB.

   a. Per ordinare la tabella, utilizza l'intestazione della colonna.

   b. Per creare il grafico di un parametro, seleziona la casella di controllo accanto al parametro. Per selezionare tutte i parametri, seleziona la casella di controllo nella riga dell'intestazione della tabella.

   c. Per filtrare per metrica, passa il mouse sul nome della metrica e seleziona la freccia del menu a discesa accanto al nome della metrica. Quindi, scegli Aggiungi alla ricerca, come mostrato nell'immagine seguente.



Using the AWS CLI

Per visualizzare CloudWatch i dati per Amazon DocumentDB, utilizza l' CloudWatch `get-metric-statistics`operazione con i seguenti parametri.

Parameters

- **`--namespace`**: obbligatorio. Lo spazio dei nomi del servizio di cui desideri visualizzare i parametri CloudWatch . Per Amazon DocumentDB, questo deve essere. `AWS/DocDB`
- **`--metric-name`**: obbligatorio. Il nome della metrica per la quale desideri i dati.
- **`--start-time`**: obbligatorio. Il timestamp che determina il primo punto dati da restituire.

  Il valore specificato è inclusivo. I risultati comprendono i punti dati con il timestamp specificato. Il timestamp deve essere in formato ISO 8601 UTC (ad esempio, 2016-10-03). T23:00:00Z

- **`--end-time`**: obbligatorio. Il timestamp che determina l'ultimo punto dati da restituire.

  Il valore specificato è inclusivo. I risultati comprendono i punti dati con il timestamp specificato. Il timestamp deve essere in formato ISO 8601 UTC (ad esempio, 2016-10-03). T23:00:00Z

- **`--period`**: obbligatorio. La granularità, in secondi, per i punti dati restituiti. Per i parametri con una risoluzione regolare, un periodo può avere un valore minimo di un minuto (60 secondi) e deve essere un multiplo di 60. Per i parametri ad alta risoluzione raccolti a intervalli inferiori al minuto, il periodo può essere 1, 5, 10, 30, 60 o qualsiasi multiplo di 60.

- **`--dimensions`**— Opzionale. Se la metrica contiene più dimensioni, è necessario includere un valore per ciascuna dimensione. CloudWatch considera ogni combinazione univoca di dimensioni come una metrica separata. Se una specifica combinazione di dimensioni non è stata pubblicata, non puoi recuperare le statistiche associate. Occorre specificare le stesse dimensioni utilizzate al momento della creazione dei parametri.

- **`--statistics`**— Opzionale. Le statistiche dei parametri diverse dai percentili.
  Per le statistiche dei percentili, utilizzare `ExtendedStatistics`. Quando chiami `GetMetricStatistics`, devi specificare `Statistics` o `ExtendedStatistics`, ma non entrambi.

  Valori consentiti:

  - `SampleCount`
  - `Average`
  - `Sum`
  - `Minimum`
  - `Maximum`

- **`--extended-statistics`**— Facoltativo. Le statistiche `percentile`. Specifica valori compresi tra p0.0 e p100. Quando chiami `GetMetricStatistics`, devi specificare `Statistics` o `ExtendedStatistics`, ma non entrambi.

- **`--unit`**— Facoltativo. L'unità per un determinato parametro. I parametri possono essere riportati con più unità. Se non specifichi un'unità, vengono restituite tutte le unità. Se specifichi solo un'unità che il parametro non supporta, i risultati della chiamata sono nulli.

  Valori possibili:

  - `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

L'esempio seguente rileva il valore massimo di CPUUtilization per un periodo di 2 ore, durante il quale viene estratto un campione ogni 60 secondi.

Per Linux, macOS o 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
```

Per 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
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
       "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"
        }
    ]
}
```

Dimensioni di Amazon DocumentDB

Le metriche per Amazon DocumentDB sono qualificate dai valori dell'account o dell'operazione. Puoi utilizzare la CloudWatch console per recuperare i dati di Amazon DocumentDB filtrati in base a una qualsiasi delle dimensioni nella tabella seguente.

Dimensione	Description
`DBClusterIdentifier`	Filtra i dati richiesti per uno specifico cluster Amazon DocumentDB.
`DBClusterIdentifier, Role`	Filtra i dati richiesti per uno specifico cluster Amazon DocumentDB, aggregando la metrica per ruolo dell'istanza (). WRITER/READER Ad esempio, puoi aggregare i parametri per tutte le istanze READER che appartengono a un cluster.
`DBInstanceIdentifier`	Filtra i dati richiesti per un'istanza database specifica.

Monitoraggio delle metriche opcounter

Le metriche opcounter hanno un valore diverso da zero (in genere ~50) per i cluster inattivi. Questo perché Amazon DocumentDB esegue controlli periodici dello stato, operazioni interne e attività di raccolta delle metriche.

Monitoraggio delle connessioni al database

Quando visualizzi il numero di connessioni utilizzando comandi del motore di database, ad esempio`db.runCommand( { serverStatus: 1 })`, potresti vedere fino a 10 connessioni in più

rispetto a quelle visualizzate. `DatabaseConnections` CloudWatch Ciò si verifica perché Amazon DocumentDB esegue controlli di integrità periodici e attività di raccolta delle metriche che non vengono prese in considerazione. `DatabaseConnections` `DatabaseConnections`rappresenta solo le connessioni avviate dal cliente.

Registrazione delle chiamate API di Amazon DocumentDB con AWS CloudTrail

Amazon DocumentDB (con compatibilità con MongoDB) è integrato con AWS CloudTrail, un servizio che fornisce un registro delle azioni intraprese da utenti, ruoli o un servizio in AWS Amazon DocumentDB (con compatibilità MongoDB). CloudTrail acquisisce tutte le chiamate AWS CLI API per Amazon DocumentDB come eventi, incluse le chiamate dalla console Amazon DocumentDB e le chiamate di codice all'SDK Amazon DocumentDB. Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per Amazon DocumentDB. Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti sulla CloudTrail console nella cronologia degli eventi. Utilizzando le informazioni raccolte da CloudTrail, puoi determinare la richiesta effettuata ad Amazon DocumentDB (compatibile con MongoDB), l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata effettuata e altri dettagli.

> **Important**
>
> Per alcune funzionalità di gestione, Amazon DocumentDB utilizza la tecnologia operativa condivisa con Amazon Relational Database Service (Amazon RDS). Le chiamate alla console e alle API di Amazon DocumentDB vengono registrate come chiamate effettuate all'API Amazon RDS. AWS CLI

Per ulteriori informazioni AWS CloudTrail, consulta AWS CloudTrail la Guida per l'utente.

Informazioni su Amazon DocumentDB in CloudTrail

CloudTrail è abilitato sul tuo account al Account AWS momento della creazione dell'account. Quando si verifica un'attività in Amazon DocumentDB (con compatibilità con MongoDB), tale attività viene registrata in un CloudTrail evento insieme ad altri AWS eventi di servizio nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare eventi recenti nel tuo. Account AWS Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi nel tuo Account AWS, inclusi gli eventi per Amazon DocumentDB (con compatibilità con MongoDB), crea un trail. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando si crea un percorso nella console, questo sarà valido in tutte le Regioni AWS. Il percorso registra gli eventi di tutte le Regioni nella partizione AWS e distribuisce i file di log nel bucket Amazon S3 specificato. Inoltre, puoi configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei CloudTrail log. Per ulteriori informazioni, consulta gli argomenti seguenti nella Guida per l'utente di AWS CloudTrail :

- Panoramica della creazione di un trail
- CloudTrail Servizi e integrazioni supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni
- Ricezione di file di CloudTrail registro da più account

Ogni evento o voce di log include informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con credenziali utente o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro AWS servizio.

Per ulteriori informazioni, consulta Elemento CloudTrail userIdentity.

Profilazione delle operazioni di Amazon DocumentDB

Puoi utilizzare il profiler in Amazon DocumentDB (con compatibilità con MongoDB) per registrare il tempo di esecuzione e i dettagli delle operazioni eseguite sul tuo cluster. Il profiler è utile per monitorare le operazioni più lente sul cluster per aiutare a migliorare le prestazioni delle singole query e le prestazioni complessive del cluster.

Per impostazione predefinita, la funzionalità profiler è disabilitata. Se abilitato, il profiler registra le operazioni che richiedono più tempo di un valore di soglia definito dal cliente (ad esempio, 100 ms) in Amazon Logs. CloudWatch I dettagli registrati includono il comando profilato, l'ora, il riepilogo del piano e i metadati del client. Dopo aver registrato le operazioni su CloudWatch Logs, puoi utilizzare

CloudWatch Logs Insights per analizzare, monitorare e archiviare i dati di profilazione di Amazon DocumentDB. Le query comuni vengono fornite nella sezione Domande comuni.

Se abilitato, il profiler utilizza risorse aggiuntive nel cluster. Inizia con un valore di soglia elevato (ad esempio 500 ms) e riduci gradualmente il valore per identificare le operazioni lente. Iniziare con un valore di soglia di 50 ms può causare problemi di prestazioni sul cluster per applicazioni a throughput elevato. Il profiler è abilitato a livello di cluster e funziona su tutte le istanze e i database di un cluster. Amazon DocumentDB registra le operazioni su Amazon CloudWatch Logs nel migliore dei modi.

Sebbene Amazon DocumentDB non imponga costi aggiuntivi per abilitare il profiler, ti vengono addebitate le tariffe standard per l'utilizzo dei log. CloudWatch Per informazioni sui prezzi di CloudWatch Logs, consulta i prezzi di Amazon. CloudWatch

Argomenti

- Operazioni supportate
- Limitazioni
- Abilitazione del profiler Amazon DocumentDB
- Disattivazione del profiler Amazon DocumentDB
- Disattivazione dell'esportazione dei log del profiler
- Accesso ai log del profiler di Amazon DocumentDB
- Domande comuni

Operazioni supportate

Il profiler Amazon DocumentDB supporta le seguenti operazioni:

- `aggregate`
- `count`
- `delete`
- `distinct`
- `find` (OP_QUERY e comando)
- `findAndModify`
- `insert`
- `update`

Limitazioni

Lo slow query profiler è in grado di emettere i log del profiler solo se l'intero set di risultati della query è in grado di rientrare in un batch e se il set di risultati è inferiore a 16 MB (dimensione massima BSON). I set di risultati superiori a 16 MB vengono suddivisi automaticamente in più batch.

La maggior parte dei driver o delle shell potrebbe impostare una dimensione batch predefinita ridotta. È possibile specificare la dimensione del batch come parte della query. Per acquisire i log delle query lente, utilizzate una dimensione del batch che superi la dimensione del set di risultati previsto. Se non sei sicuro della dimensione del set di risultati o se varia, imposta la dimensione del batch su un numero elevato (ad esempio, 100k).

Tuttavia, l'utilizzo di un batch di dimensioni maggiori significa che sarà necessario recuperare più risultati dal database prima di inviare una risposta al client. Per alcune domande, ciò potrebbe comportare ritardi più lunghi prima di ottenere risultati. Se non si prevede di utilizzare l'intero set di risultati, è possibile che si spenda di più I/Os per elaborare la query e buttare via il risultato.

Abilitazione del profiler Amazon DocumentDB

L'abilitazione del profiler su un cluster è un processo in tre fasi. Assicurati che tutti i passaggi siano completati, altrimenti i log di profilazione non verranno inviati a CloudWatch Logs. Il profiler è impostato a livello di cluster e viene eseguito su tutti i database e le istanze del cluster.

Per abilitare il profiler su un cluster

1. Poiché non puoi modificare un gruppo predefinito di parametri del cluster, assicurati di disporre di un gruppo personalizzato di parametri del cluster. Per ulteriori informazioni, consulta Creazione di gruppi di parametri del cluster Amazon DocumentDB.
2. Utilizzando un gruppo di parametri personalizzato del cluster disponibile, modifica i seguenti parametri: `profiler`, `profiler_threshold_ms`, e `profiler_sampling_rate`. Per ulteriori informazioni, consulta Modifica dei gruppi di parametri del cluster Amazon DocumentDB.
3. Crea o modifica il cluster per utilizzare il gruppo di parametri del cluster personalizzato e abilitare l'esportazione dei `profiler` log in Logs. CloudWatch

Le sezioni seguenti mostrano come implementare questi passaggi utilizzando Console di gestione AWS and the AWS Command Line Interface (AWS CLI).

Using the Console di gestione AWS

1. Prima di iniziare, crea un cluster Amazon DocumentDB e un gruppo di parametri del cluster personalizzato, se non ne hai già uno. Per ulteriori informazioni, consultare Creazione di gruppi di parametri del cluster Amazon DocumentDB e Creazione di un cluster Amazon DocumentDB.
2. Utilizzando un gruppo personalizzato di parametri del cluster disponibile, modifica i parametri seguenti. Per ulteriori informazioni, consulta Modifica dei gruppi di parametri del cluster Amazon DocumentDB.

   - `profiler`— Abilita o disabilita la profilazione delle query. I valori consentiti sono `enabled` e `disabled`. Il valore predefinito è `disabled`. Per abilitare il profiling, impostare il valore su `enabled`.
   - `profiler_threshold_ms`— Quando `profiler` è impostato su`enabled`, tutti i comandi che richiedono più tempo di quanto `profiler_threshold_ms` vengono registrati. CloudWatch I valori consentiti sono `[50-INT_MAX]`. Il valore predefinito è `100`.
   - `profiler_sampling_rate`— La frazione di operazioni lente che devono essere profilate o registrate. I valori consentiti sono `[0.0-1.0]`. Il valore predefinito è `1.0`.
3. Modifica il cluster per utilizzare il gruppo di parametri del cluster personalizzato e imposta le esportazioni dei log del profiler da pubblicare su Amazon. CloudWatch

   a. Nel riquadro di navigazione scegliere Clusters (Cluster) per aggiungere il nuovo gruppo di parametri personalizzato a un cluster.

   b. Scegliere il pulsante a sinistra del nome del cluster a cui si desidera associare il gruppo di parametri. Selezionare Actions (Operazioni), quindi Modify (Modifica) per modificare il cluster.

   c. In Cluster options (Opzioni cluster), scegliere il gruppo di parametri personalizzato dal passaggio precedente per aggiungerlo al cluster.

   d. In Log exports, seleziona Profiler logs da pubblicare su Amazon. CloudWatch

   e. Scegliere Continue (Continua) per visualizzare un riepilogo delle modifiche.

   f. Dopo aver verificato le modifiche, è possibile applicarle immediatamente o durante la successiva finestra di manutenzione in Scheduling of modifications (Pianificazione delle modifiche).

   g. Scegliere Modify cluster (Modifica cluster) per aggiornare il cluster con il nuovo gruppo di parametri.

Using the AWS CLI

La procedura seguente abilita il profiler su tutte le operazioni supportate per il cluster `sample-cluster`.

1. Prima di iniziare, assicurati di avere un gruppo di parametri del cluster personalizzati disponibile eseguendo il comando seguente e verificando che l'output di un gruppo di parametri del cluster non abbia `default` nel nome e che abbia `docdb3.6` come famiglia del gruppi di parametri. Se non disponi di un gruppo di parametri cluster non predefinito, consulta Creazione di gruppi di parametri del cluster Amazon DocumentDB.

```
aws docdb describe-db-cluster-parameter-groups \
    --query 'DBClusterParameterGroups[*].
[DBClusterParameterGroupName,DBParameterGroupFamily]'
```

   Nel seguente output, solo `sample-parameter-group` soddisfa entrambi i criteri.

```
[
       [
           "default.docdb3.6",
           "docdb3.6"
       ],
       [
           "sample-parameter-group",
           "docdb3.6"
       ]
]
```

2. Utilizzando il gruppo personalizzato di parametri del cluster, modifica i parametri seguenti:

   - `profiler`— Abilita o disabilita la profilazione delle query. I valori consentiti sono `enabled` e `disabled`. Il valore predefinito è `disabled`. Per abilitare il profiling, impostare il valore su `enabled`.
   - `profiler_threshold_ms`— Quando `profiler` è impostato su`enabled`, tutti i comandi che richiedono più tempo di quanto `profiler_threshold_ms` vengono registrati. CloudWatch I valori consentiti sono `[50-INT_MAX]`. Il valore predefinito è `100`.
   - `profiler_sampling_rate`— La frazione di operazioni lente che devono essere profilate o registrate. I valori consentiti sono `[0.0-1.0]`. Il valore predefinito è `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. Modifica il cluster Amazon DocumentDB in modo che utilizzi il gruppo di parametri del cluster `sample-parameter-group` personalizzato del passaggio precedente e imposti il parametro su. `--enable-cloudwatch-logs-exports` `profiler`

   Il codice seguente modifica il cluster `sample-cluster` per utilizzare il `sample-parameter-group` passaggio precedente e si aggiunge `profiler` alle esportazioni di CloudWatch Logs abilitate.

   ```
   aws docdb modify-db-cluster \
          --db-cluster-identifier sample-cluster \
          --db-cluster-parameter-group-name sample-parameter-group \
          --cloudwatch-logs-export-configuration '{"EnableLogTypes":["profiler"]}'
   ```

   L'aspetto dell'output di questa operazione è simile al seguente.

   ```
   {
       "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
    }
}
```

Disattivazione del profiler Amazon DocumentDB

Per disabilitare il profiler, disabiliti sia il `profiler` parametro che l'esportazione dei `profiler` log in Logs. CloudWatch

Disabilitazione del profiler

È possibile disattivare il `profiler` parametro utilizzando Console di gestione AWS o AWS CLI, come segue.

Using the Console di gestione AWS

La seguente procedura utilizza Amazon DocumentDB Console di gestione AWS per disabilitare Amazon DocumentDB. `profiler`

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri). Scegliere quindi il nome del gruppo di parametri del cluster su cui si desidera disabilitare il profiler.
3. Nella pagina Cluster parameters (Parametri cluster) risultante, selezionare il pulsante a sinistra del parametro `profiler` e scegliere Edit (Modifica).
4. Nella finestra di dialogo Modify profiler (Modifica profiler), scegliere `disabled` nell'elenco.
5. Scegliere Modify cluster parameter (Modifica parametro cluster).

Using the AWS CLI

Per disabilitare `profiler` su un cluster utilizzando l' AWS CLI, modificare il cluster come segue.

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 ParameterName=profiler,ParameterValue=disabled,ApplyMethod=immediate
```

Disattivazione dell'esportazione dei log del profiler

È possibile disabilitare l'esportazione `profiler` dei CloudWatch log in Logs utilizzando Console di gestione AWS o AWS CLI, come segue.

Using the Console di gestione AWS

La seguente procedura utilizza Console di gestione AWS per disabilitare l'esportazione dei log in Amazon DocumentDB. CloudWatch

1. Apri la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster. Scegliere il pulsante a sinistra del nome del cluster per il quale si desidera disabilitare l'esportazione dei log.
3. Nel menu Actions (Operazioni) selezionare Modify (Modifica).
4. Scorrere verso il basso fino alla sezione Log exports (Esportazioni log) e deselezionare Profiler logs (Log del profiler).
5. Scegli Continua.
6. Esaminare le modifiche, quindi scegliere quando applicare la modifica al cluster.
   - Apply during the next scheduled maintenance window (Applica durante la prossima finestra di manutenzione pianificata)
   - Apply immediately (Applica immediatamente)
7. Scegliere Modify cluster (Modifica cluster).

Using the AWS CLI

Il codice seguente modifica il cluster `sample-cluster` e disabilita CloudWatch i log del profiler.

Example

Per Linux, macOS o Unix:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["profiler"]}'
```

Per Windows:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["profiler"]}'
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
{
    "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
    }
}
```

Accesso ai log del profiler di Amazon DocumentDB

Segui questi passaggi per accedere ai log del tuo profilo su Amazon CloudWatch.

1. Apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Assicurati di trovarti nella stessa regione del cluster Amazon DocumentDB.
3. Nel riquadro di navigazione scegli Logs (Log).
4. Per trovare i log del profiler per il cluster, nell'elenco scegliere `/aws/docdb/`*`yourClusterName`*`/profiler`.

   I log del profiler per ogni istanza sono disponibili nei rispettivi nomi di istanza.

Domande comuni

Di seguito sono riportate alcune query comuni che è possibile utilizzare per analizzare i comandi profilati. Per ulteriori informazioni su CloudWatch Logs Insights, vedi Analisi dei dati di CloudWatch log con Logs Insights e Sample Queries.

Ottieni le 10 operazioni più lente su una raccolta specificata

```
filter ns="test.foo" | sort millis desc | limit 10
```

Ottieni tutte le operazioni di aggiornamento su una raccolta che ha richiesto più di 60 ms

```
filter millis > 60 and op = "update"
```

Ottieni le 10 operazioni più lente dell'ultimo mese

```
sort millis desc | limit 10
```

Ottenere tutte le query con un riepilogo del piano COLLSCAN

```
filter planSummary="COLLSCAN"
```

Monitoraggio con Performance Insights

Performance Insights si aggiunge alle funzionalità di monitoraggio esistenti di Amazon DocumentDB per illustrare le prestazioni del cluster e aiutarti ad analizzare eventuali problemi che lo riguardano. Con la dashboard di Performance Insights, puoi visualizzare il carico del database e filtrare il carico per attese, istruzioni di interrogazione, host o applicazione.

> **Note**
>
> Performance Insights è disponibile solo per i cluster basati su istanze Amazon DocumentDB 3.6, 4.0, 5.0 e 8.0.

In che modo è utile?

- Visualizza le prestazioni del database: visualizza il carico per determinare quando e dove si trova il carico sul database
- Determina la causa del carico sul database: determina quali query, host e applicazioni contribuiscono al carico sull'istanza
- Determina quando c'è carico sul tuo database: ingrandisci la dashboard di Performance Insights per concentrarti su eventi specifici o riduci lo zoom per osservare le tendenze in un arco di tempo più ampio
- Avviso sul carico del database: accedi automaticamente alle nuove metriche di carico del database da CloudWatch cui puoi monitorare le metriche di carico del database insieme ad altre metriche di Amazon DocumentDB e impostare avvisi su di esse

Quali sono i limiti di Amazon DocumentDB Performance Insights?

- Gli approfondimenti sulle prestazioni nelle regioni AWS GovCloud AWS GovCloud (US-EastUS-West) e () non sono disponibili
- Performance Insights per Amazon DocumentDB conserva fino a 7 giorni di dati sulle prestazioni
- Le query più lunghe di 1.024 byte non vengono aggregate in Performance Insights

Argomenti

- Concetti relativi a Performan
- Abilitazione e disattivazione di Performance Insights
- Configurazione delle policy di accesso per Performance Insights
- Per analizzare il parametro utilizzando il pannello di controllo di Performance Insights
- Recupero dei parametri con l'API Performance Insights
- CloudWatch Metriche di Amazon per Performance Insights
- Performance Insights per le contrometriche

Concetti relativi a Performan

Argomenti

- Media delle sessioni attive
- Dimensioni
- Max vCPU

Media delle sessioni attive

Il carico del database (carico DB)misura il livello di attività nel database. Il parametro chiave in Performance Insights è `DB Load`, che viene raccolto ogni secondo. L'unità di misura è l'`DBLoad`Average Active Sessions (AAS) per un'istanza Amazon DocumentDB.

Una sessione attiva è una connessione che ha inviato il lavoro all'istanza Amazon DocumentDB ed è in attesa di una risposta. Ad esempio, se invii una query a un'istanza di Amazon DocumentDB, la sessione del database è attiva mentre l'istanza sta elaborando la query.

Per ottenere le sessioni attive medie (AAS), Performance Insights esegue il campionamento del numero di sessioni che eseguono contemporaneamente una query. L'AAS è il numero totale

di sessioni diviso per il numero totale di campioni. La tabella seguente mostra cinque esempi consecutivi di una query in esecuzione.

Project N.E.M.O.	Numero di sessioni che eseguono query	AAS	Calcolo
1	2	2	2 sessioni/1 campione
2	0	1	2 sessioni/2 campioni
3	4	2	6 sessioni/3 campioni
4	0	1.5	6 sessioni/4 campioni
5	4	2	10 sessioni/5 campioni

Nell'esempio precedente, il carico del database per l'intervallo di tempo compreso tra 1 e 5 è 2 AAS. Un aumento del carico database significa che, in media, più sessioni sono in esecuzione sul database.

Dimensioni

Il parametro DB Load è diverso dagli altri parametri di serie temporali in quanto può essere suddiviso in sottocomponenti detti dimensioni. Le dimensioni possono essere considerate come categorie per le diverse caratteristiche della metrica DB Load. Quando si diagnosticano problemi di prestazioni, le dimensioni più utili sono gli stati di attesa e la top query.

stati di attesa

Uno stato di attesa fa sì che un'istruzione di query attenda il verificarsi di un evento specifico prima di poter continuare a funzionare. Ad esempio, un'istruzione di interrogazione potrebbe attendere lo sblocco di una risorsa bloccata. Combinandolo DB Load con gli stati di attesa, è possibile ottenere un quadro completo dello stato della sessione. Ecco i vari stati di attesa di Amazon DocumentDB:

Stato di attesa di Amazon DocumentDB	Descrizione dello stato di attesa
Chiusura	Lo stato di attesa Latch si verifica quando la sessione è in attesa di paginare il buffer

Stato di attesa di Amazon DocumentDB	Descrizione dello stato di attesa
	pool. L'impaginazione frequente in entrata e in uscita dal buffer pool può verificarsi più spesso quando il sistema elabora frequentemente interrogazioni di grandi dimensioni, scansioni di raccolte o quando il pool di buffer è troppo piccolo per gestire il working set.
CPU	Lo stato di attesa della CPU si verifica quando la sessione è in attesa sulla CPU.
CollectionLock	Lo stato di CollectionLock attesa si verifica quando la sessione è in attesa di acquisire un blocco sulla raccolta. Questi eventi si verifican o quando sono presenti operazioni DDL sulla raccolta.
DocumentLock	Lo stato di DocumentLock attesa si verifica quando la sessione è in attesa di acquisire un blocco su un documento. Un numero elevato di scritture simultanee sullo stesso documento contribuirà a creare più stati di DocumentLock attesa su quel documento.
SystemLock	Lo stato di SystemLock attesa si verifica quando la sessione è in attesa nel sistema. Ciò può verificarsi quando nel sistema sono presenti frequenti interrogazioni di lunga durata, transazioni di lunga durata o elevata concorren za.
IO	Lo stato di attesa IO si verifica quando la sessione è in attesa del completamento dell'IO.

Stato di attesa di Amazon DocumentDB	Descrizione dello stato di attesa
BufferLock	Lo stato di BufferLock attesa si verifica quando la sessione è in attesa di acquisire un blocco su una pagina condivisa nel buffer. BufferLockgli stati di attesa possono essere prolungati se altri processi tengono aperti i cursori sulle pagine richieste.
LowMemThrottle	Lo stato di LowMemThrottle attesa si verifica quando la sessione è in attesa a causa della forte pressione di memoria sull'istanza Amazon DocumentDB. Se questo stato persiste per un lungo periodo, valuta la possibilità di scalare l'istanza per fornire memoria aggiuntiva. Per ulteriori informazioni, consulta Resource Governor.
BackgroundActivity	Lo stato di BackgroundActivity attesa si verifica quando la sessione è in attesa di processi interni del sistema.
Altro	L'altro stato di attesa è uno stato di attesa interno. Se questo stato persiste per un lungo periodo, valuta la possibilità di terminare questa query. Per ulteriori informazioni, vedi Come posso trovare e terminare le query bloccate o di lunga durata?

Domande principali

Mentre gli stati di attesa mostrano dei colli di bottiglia, le prime query mostrano quali sono le query che contribuiscono maggiormente al carico del DB. Ad esempio, molte query potrebbero essere attualmente in esecuzione nel database, ma una singola query potrebbe consumare il 99% del carico DB. In questo caso, il carico elevato potrebbe indicare un problema con la query.

Max vCPU

Nel dashboard, il grafico di caricamento del database raccoglie, aggrega e visualizza le informazioni sulla sessione. Per verificare se le sessioni attive superano la CPU massima, esaminare la loro relazione con la linea vCPU massima. Il valore vCPU massimo è determinato dal numero di core vCPU (CPU virtuale) per l'istanza Amazon DocumentDB.

Se il carico è spesso sopra la linea vCPU massima e lo stato di attesa primario è CPU, la CPU è sovraccarica. In questo caso, potresti voler limitare le connessioni all'istanza, ottimizzare le query con un carico di CPU elevato o prendere in considerazione una classe di istanza più ampia. Istanze elevate e costanti di qualsiasi stato di attesa indicano che possono verificarsi colli di bottiglia o problemi di conflitto delle risorse da risolvere. Questo può valere anche se il carico database non supera il valore della riga CPU massima.

Abilitazione e disattivazione di Performance Insights

Per utilizzare Performance Insights, è necessario attivarlo nell'istanza database. Puoi disabilitarlo in un secondo momento, se necessario. L'attivazione e la disattivazione di Performance Insights non determina tempi di inattività, riavvio o failover.

L'agente Performance Insights consuma CPU e memoria limitate sull'host DB. Quando il carico del DB è elevato, l'agente limita l'impatto sulle prestazioni raccogliendo i dati meno frequentemente.

Abilitazione di Performance Insights durante la creazione di un cluster

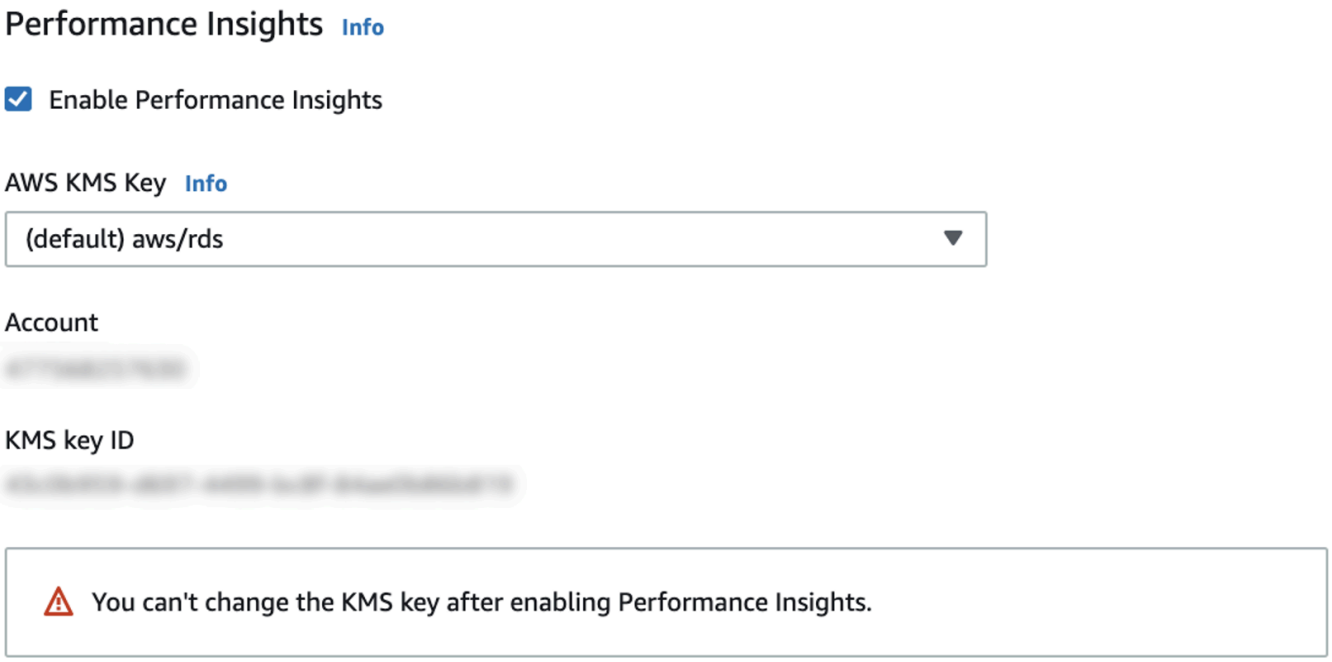
Nella console è possibile attivare o disattivare Performance Insights quando si crea o si modifica una nuova istanza DB.

Utilizzo di Console di gestione AWS

Nella console, puoi abilitare Performance Insights quando crei un cluster Amazon DocumentDB. Quando crei un nuovo cluster Amazon DocumentDB, abilita Performance Insights scegliendo Abilita Performance Insights nella sezione Performance Insights.

Istruzioni della console

1. Per creare un cluster, segui le istruzioni per la creazione di un cluster Amazon DocumentDB.
2. Seleziona Abilita Performance Insights nella sezione Performance Insights.



> **Note**
>
> Il periodo di conservazione dei dati di Performance Insights sarà di sette giorni.

AWS KMS chiave: specifica la tua chiave AWS KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili utilizzando la tua AWS KMS chiave. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Configurazione di una AWS AWS KMS policy per Performance Insights.

Abilitazione e disabilitazione durante la modifica di un'istanza

È possibile modificare un'istanza DB per abilitare o disabilitare Performance Insights utilizzando la console o. AWS CLI

Using the Console di gestione AWS

Istruzioni della console

1. Accedi e apri Console di gestione AWS la console di Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.
2. Scegli Cluster.

3. Scegliere un'istanza DB e scegliere Modifica.
4. Nella sezione Performance Insights, scegli Abilita Performance Insights o Disabilita Performance Insights.

   > **Note**
   >
   > Se scegli Abilita Performance Insights, puoi specificare la tua AWS AWS KMS chiave. Performance Insights crittografa tutti i dati potenzialmente sensibili utilizzando la tua AWS KMS chiave. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Encrypting Amazon DocumentDB Data at Rest.

5. Scegli Continue (Continua).
6. In Scheduling of Modifications (Pianificazione delle modifiche), scegliere Apply immediately (Applica immediatamente). Se scegli Applica durante la successiva finestra di manutenzione programmata, l'istanza ignora questa impostazione e abilita immediatamente Performance Insights.
7. Scegli Modify instance (Modifica istanza).

Using the AWS CLI

Quando utilizzi i `modify-db-instance` AWS AWS CLI comandi `create-db-instance` or, puoi abilitare Performance Insights `--enable-performance-insights` specificandolo o disabilitarlo specificando. `--no-enable-performance-insights`

La seguente procedura descrive come attivare Performance Insights per un'istanza database utilizzando AWS AWS CLI.

AWS AWS CLI istruzioni

Chiamate il `modify-db-instance` AWS AWS CLI comando e fornite i seguenti valori:

- `--db-instance-identifer`— Il nome dell'istanza DB
- `--enable-performance-insights` per abilitare o `--no-enable-performance-insights` per disabilitare

Example

L'esempio seguente abilita Performance Insights per`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
```

Configurazione delle policy di accesso per Performance Insights

Per accedere a Performance Insights, devi avere le autorizzazioni appropriate da AWS Identity and Access Management (IAM). Per concedere l'accesso sono disponibili le seguenti opzioni:

- Collega la policy gestita `AmazonRDSPerformanceInsightsReadOnly` a un set di autorizzazioni o a un ruolo.
- Crea una policy IAM personalizzata e collegala a un set di autorizzazioni o un ruolo.

Inoltre, se è stato specificata una chiave gestita dal cliente durante l'attivazione di Performance Insights, è necessario assicurarsi che gli utenti dell'account dispongano delle autorizzazioni `kms:Decrypt` e `kms:GenerateDataKey` sulla chiave KMS.

> **Note**
>
> Per la crittografia a riposo con la gestione di AWS KMS chiavi e gruppi di sicurezza, Amazon DocumentDB sfrutta la tecnologia operativa condivisa con Amazon RDS. https://aws.amazon.com/rds

Allegare la policy a un responsabile IAM AmazonRDSPerformanceInsightsReadOnly

`AmazonRDSPerformanceInsightsReadOnly`è una policy AWS gestita che garantisce l'accesso a tutte le operazioni di sola lettura dell'API Amazon DocumentDB Performance Insights. Al momento, tutte le operazioni in questa API sono di sola lettura. Se si collega

`AmazonRDSPerformanceInsightsReadOnly` a un set di autorizzazioni o un ruolo, il destinatario può utilizzare Performance Insights insieme ad altre funzionalità della console.

Creazione di una policy IAM personalizzata per Approfondimenti sulle prestazioni

Per gli utenti che non hanno accesso completo alla policy `AmazonRDSPerformanceInsightsReadOnly`, si può concedere l'accesso a Performance Insights creando o modificando una policy IAM gestita dall'utente. Quando alleghi la policy a un set di autorizzazioni o a un ruolo, il destinatario può utilizzare Performance Insights.

Per creare una policy personalizzata

1. Aprire la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione, scegli Policy.
3. Scegli Create Policy (Crea policy).
4. Nella pagina Create Policy (Crea policy), seleziona la scheda JSON.
5. Copia e incolla il testo seguente, sostituendolo *`us-east-1`* con il nome della tua AWS regione e *`111122223333`* con il numero del tuo account cliente.

   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. Scegliere Esamina policy.
7. Specifica un nome per la policy e, facoltativamente, una descrizione e quindi scegli Crea policy.

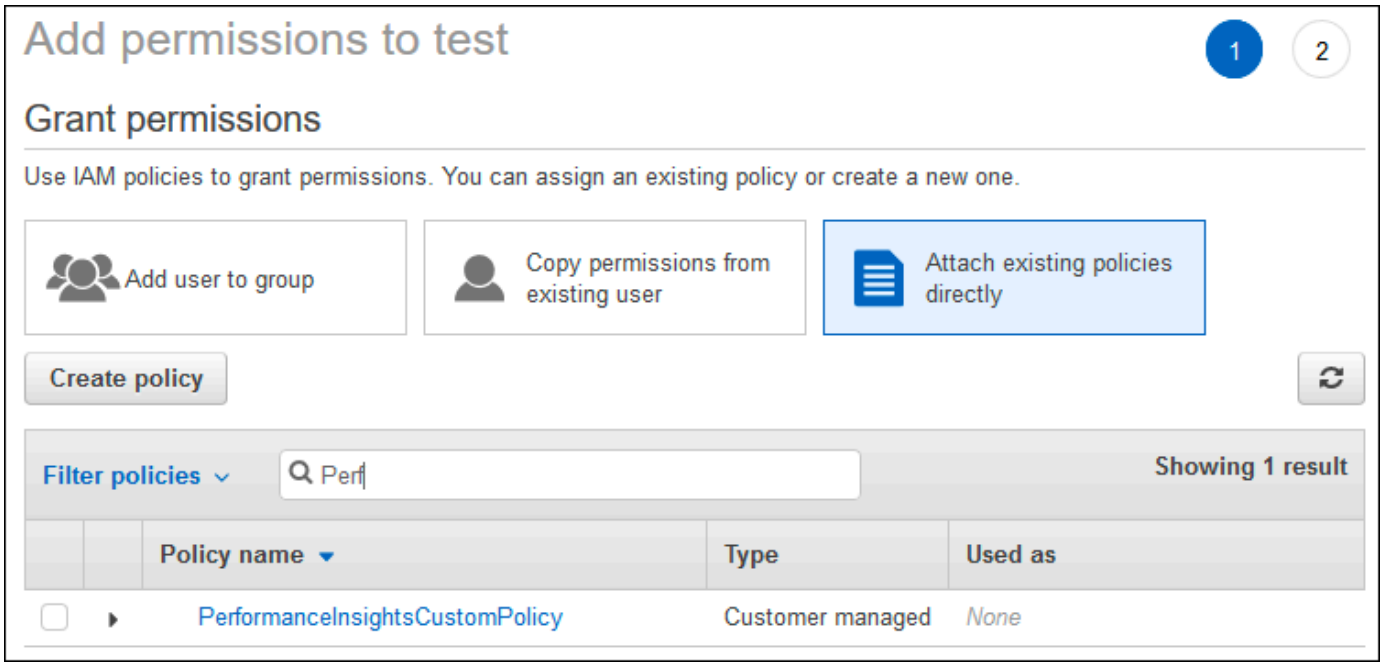
Ora è possibile collegare la policy a un set di autorizzazioni o un ruolo. La seguente procedura presuppone che si disponga già di un utente disponibile allo scopo.

Per collegare la policy a un utente

1. Aprire la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, seleziona Utenti.
3. Seleziona un utente esistente dall'elenco.

> **Important**
>
> Per utilizzare Performance Insights, assicurati di avere accesso ad Amazon DocumentDB oltre alla policy personalizzata. Ad esempio, la policy AmazonDocDBReadOnlyAccess predefinita fornisce l'accesso in sola lettura ad Amazon. DocDB.For Ulteriori informazioni, vedi Gestione dell'accesso tramite politiche.

4. Nella pagina Summary (Riepilogo), scegli Add permissions (Aggiungi autorizzazioni).
5. Scegli Attach existing policies directly (Collega direttamente le policy esistenti). Per Search (Ricerca) digita i primi caratteri del nome della policy, come mostrato di seguito.



6. Scegli la policy e quindi seleziona Next: Review (Successivo: Rivedi).
7. Scegli Add Permissions (Aggiungi autorizzazioni).

Configurazione di un AWS KMS politica per Performance Insights

Performance Insights utilizza un AWS KMS key per crittografare i dati sensibili. Quando si abilita Performance Insights mediante l'API o la console, sono disponibili le seguenti opzioni:

- Scegli l'impostazione predefinita Chiave gestita da AWS.

Amazon DocumentDB utilizza il Chiave gestita da AWS per la tua nuova istanza DB. Amazon DocumentDB crea un account Chiave gestita da AWS per il tuo AWS account. Il tuo AWS account è diverso Chiave gestita da AWS per Amazon DocumentDB per ogni AWS regione.

- Scegli una chiave gestita dal cliente.

  Se si specifica una chiave gestita dal cliente, gli utenti dell'account che chiamano l'API Performance Insights necessitano delle autorizzazioni `kms:Decrypt` e `kms:GenerateDataKey` per la chiave KMS. Gestisci le autorizzazioni della chiave KMS utilizzando la AWS KMS policy chiave anziché utilizzare le politiche IAM. Per ulteriori informazioni, vedere Utilizzo delle politiche chiave in AWS KMS.

Example

La seguente policy chiave di esempio mostra come aggiungere istruzioni alla propria policy della chiave KMS. Queste istruzioni consentono l'accesso a Performance Insights. A seconda di come utilizzi il AWS KMS, potresti voler modificare alcune restrizioni. Prima di aggiungere istruzioni alle policy, ai criteri, rimuovi tutti i commenti.

Per analizzare il parametro utilizzando il pannello di controllo di Performance Insights

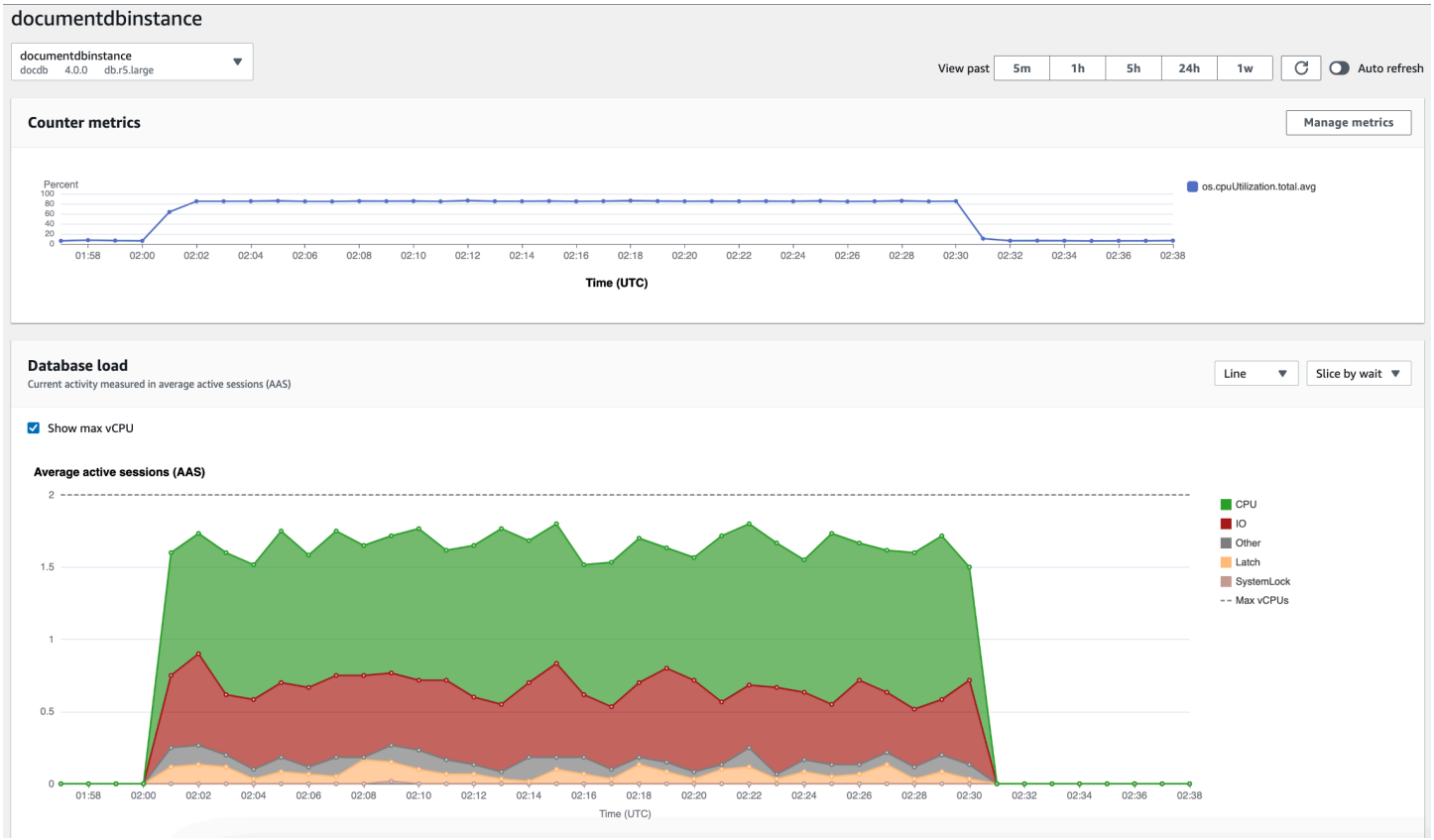
Il pannello di controllo di Performance Insights contiene informazioni sulle performance del database, per consentire di analizzare e risolvere i problemi di performance. Nella pagina principale del dashboard, puoi visualizzare le informazioni sul carico del database (carico del DB). È possibile «suddividere» il carico del DB in base a dimensioni quali stati di attesa o query.

Argomenti

- Panoramica del pannello di controllo di Performance Insights
- Per aprire il pannello di controllo di Performance Insights
- Analisi del carico del database in base agli stati di attesa
- Panoramica della scheda Principali domande
- Ingrandimento del diagramma di carico del database

Panoramica del pannello di controllo di Performance Insights

Il pannello di controllo è il modo più semplice per interagire con Performance Insights. L'esempio seguente mostra la dashboard per un'istanza Amazon DocumentDB. Per impostazione predefinita, il pannello di controllo di Performance Insights mostra i dati relativi agli ultimi 60 minuti.



Il pannello di controllo è diviso nelle seguenti parti:

1. Metriche dei contatori: mostra i dati relativi a specifiche metriche relative ai contatori delle prestazioni.
2. Caricamento del database: mostra il confronto tra il carico del database e la capacità dell'istanza database rappresentata dalla riga Max vCPU.
3. Dimensioni principali: mostra le dimensioni principali che contribuiscono al carico del DB. Queste dimensioni includono `waitsqueries`,`hosts`,`databases`, `eapplications`.

Argomenti

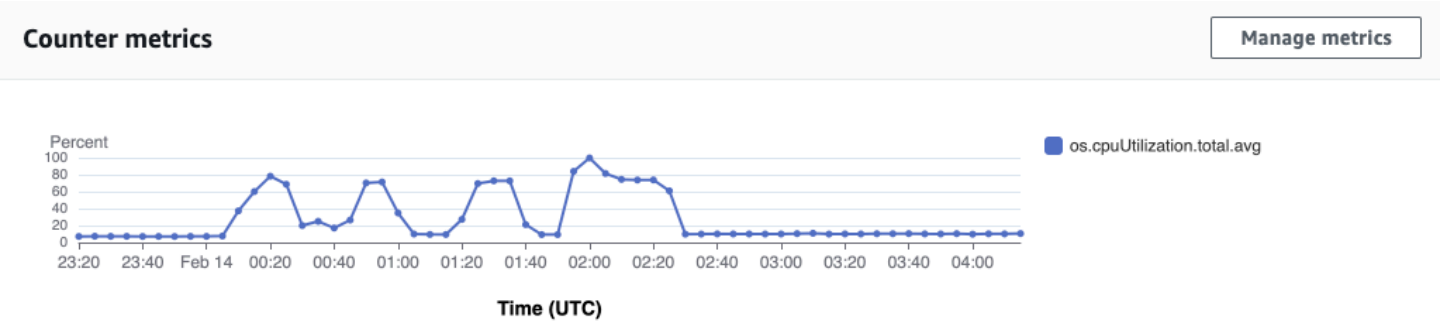
- Grafico Parametri contatore
- Grafico di carico database

- Tabella dimensioni superiori

Grafico Parametri contatore

Con i parametri contatore, puoi personalizzare il pannello di controllo di Performance Insights per includere fino a 10 grafici aggiuntivi. Questi grafici mostrano una selezione di dozzine di metriche del sistema operativo. Queste informazioni possono essere correlate ai carichi dei database per agevolare l'individuazione e l'analisi di problemi legati alle prestazioni.

Il grafico Counter Metrics (Parametri contatore) visualizza i dati per i contatori delle prestazioni.



Per modificare i contatori delle prestazioni, scegli Gestisci metriche. Puoi selezionare più metriche del sistema operativo come mostrato nella schermata seguente. Per visualizzare i dettagli relativi a qualsiasi metrica, passare il mouse sul nome della metrica.

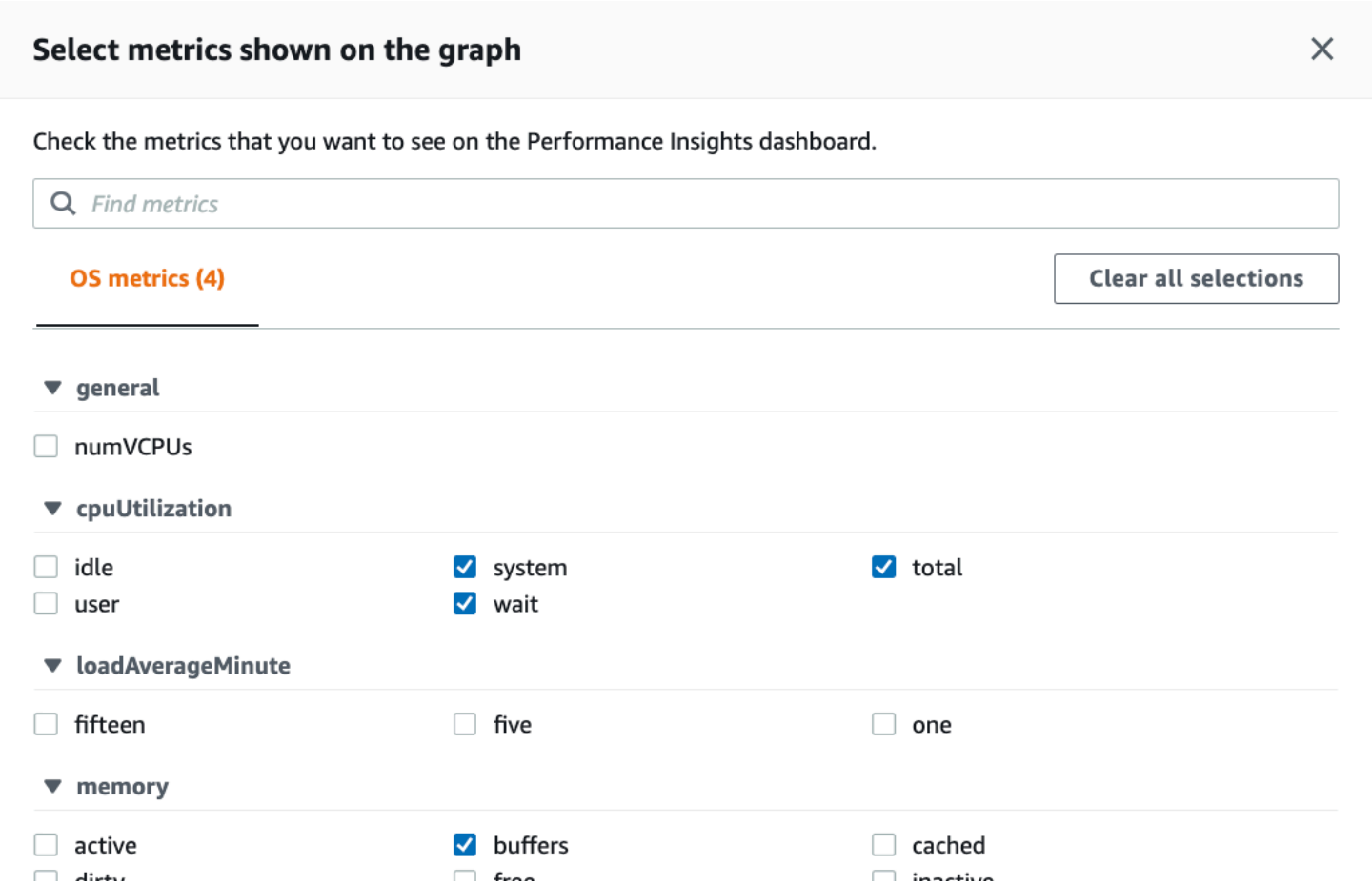


Grafico di carico database

Il grafico di caricamento del database mostra il confronto tra l'attività del database e la capacità dell'istanza rappresentata dalla riga Max vCPU. Per impostazione predefinita, il grafico a linee in pila rappresenta il carico DB come sessioni attive medie per unità di tempo. Il carico DB viene suddiviso (raggruppato) in base agli stati di attesa.



Carico del database suddiviso per dimensioni

È possibile scegliere di visualizzare il carico sotto forma di sessioni attive raggruppate in base alle dimensioni supportate. L'immagine seguente mostra le dimensioni dell'istanza Amazon DocumentDB.



Dettagli del carico DB per un elemento della dimensione

Per visualizzare i dettagli su un elemento del carico del database all'interno di una dimensione, passa il mouse sul nome dell'elemento. L'immagine seguente mostra i dettagli di una dichiarazione di interrogazione.



Per visualizzare i dettagli relativi a qualsiasi elemento per il periodo di tempo selezionato nella legenda, passa il mouse su tale elemento.

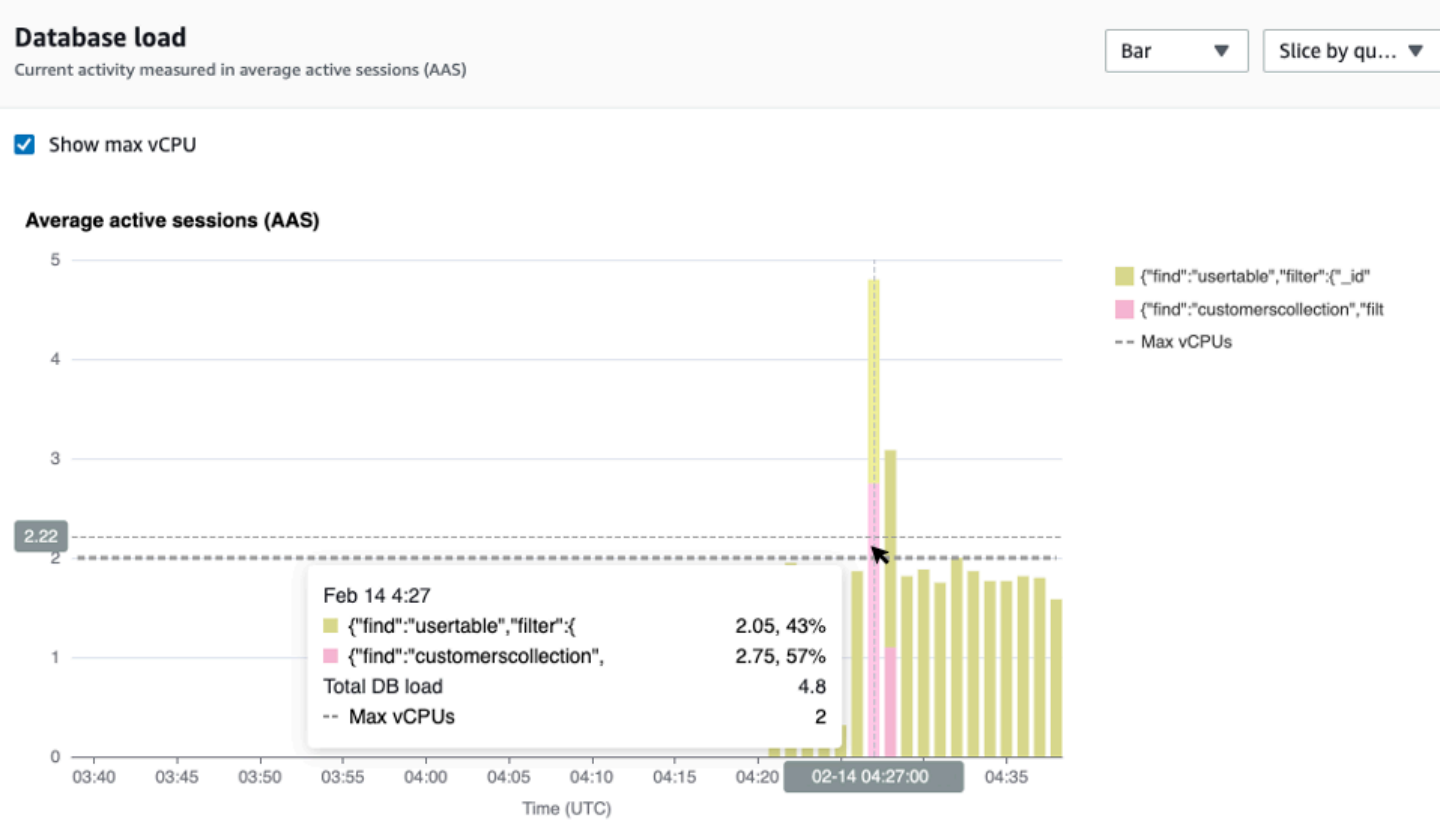
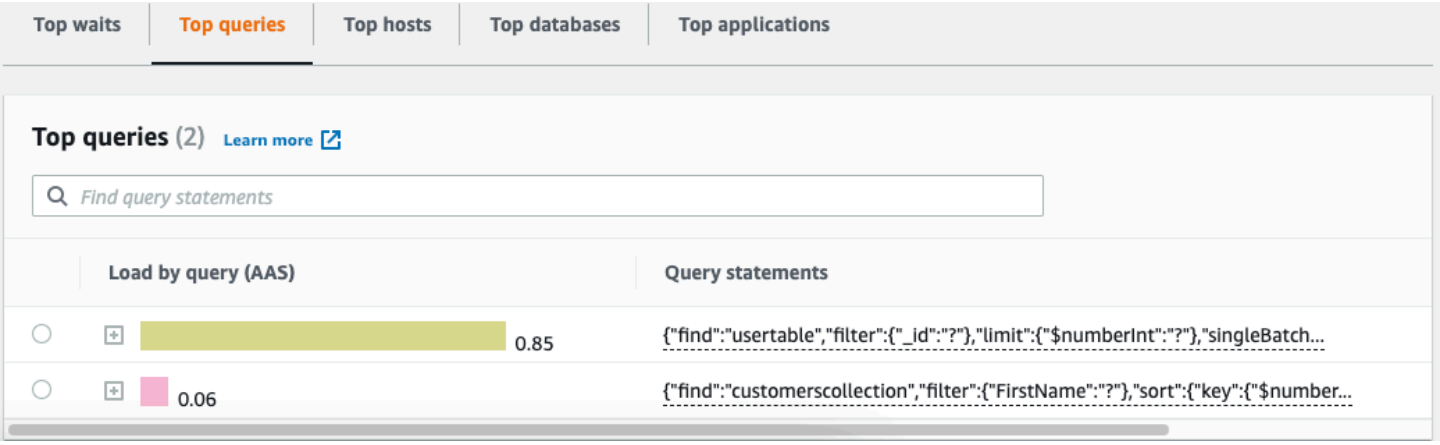


Tabella dimensioni superiori

La tabella Dimensioni principali suddivide il carico del DB in base a dimensioni diverse. Una dimensione è una categoria o una suddivisione per le diverse caratteristiche del carico del database. Se la dimensione è una query, Top queries mostra le istruzioni della query che contribuiscono maggiormente al carico del DB.

Scegli una delle seguenti schede di dimensione.



La tabella seguente fornisce una breve descrizione di ciascuna scheda.

Scheda Sessioni	Selezionati
Principali per stati di attesa	Elenco dei principali stati di attesa per il back-end del database è in attesa
Principali istruzioni di interrogazione	Domande principali di interrogazione attualmente in esecuzione
Host principali	L'IP host e la porta del client connesso

Scheda	Descrizione
Database	Database principali a cui è connesso il client
Applicazioni	Applicazioni o nomi delle applicazioni connessa al database

Per informazioni su come analizzare le interrogazioni utilizzando la scheda Principali domande, vedere. Panoramica della scheda Principali domande

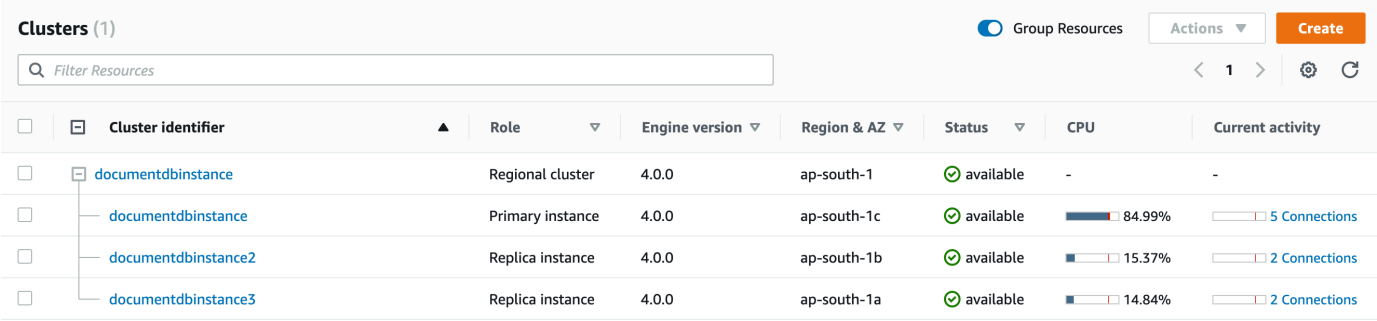
Per aprire il pannello di controllo di Performance Insights

Per visualizzare la dashboard di Performance Insights nel AWS Console di gestione, utilizza i seguenti passaggi:

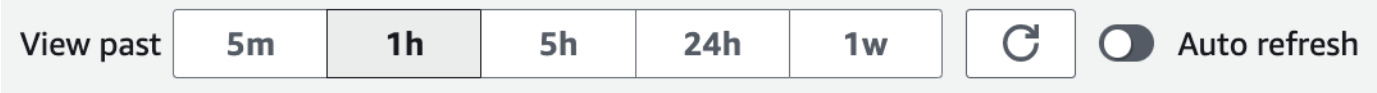
1. Apri la console Performance Insights all'indirizzo https://console.aws.amazon.com/docdb/.
2. Scegli un'istanza database. La dashboard di Performance Insights viene mostrata per quell'istanza di Amazon DocumentDB.

   Per le istanze Amazon DocumentDB con Performance Insights abilitato, puoi anche accedere alla dashboard scegliendo la voce Sessions nell'elenco delle istanze. In Attività corrente la voce Sessioni mostra il carico del database in sessioni attive medie negli ultimi cinque minuti. Il grafico mostra graficamente il carico: Quando la barra è vuota, l'istanza è inattiva. Con l'aumentare del

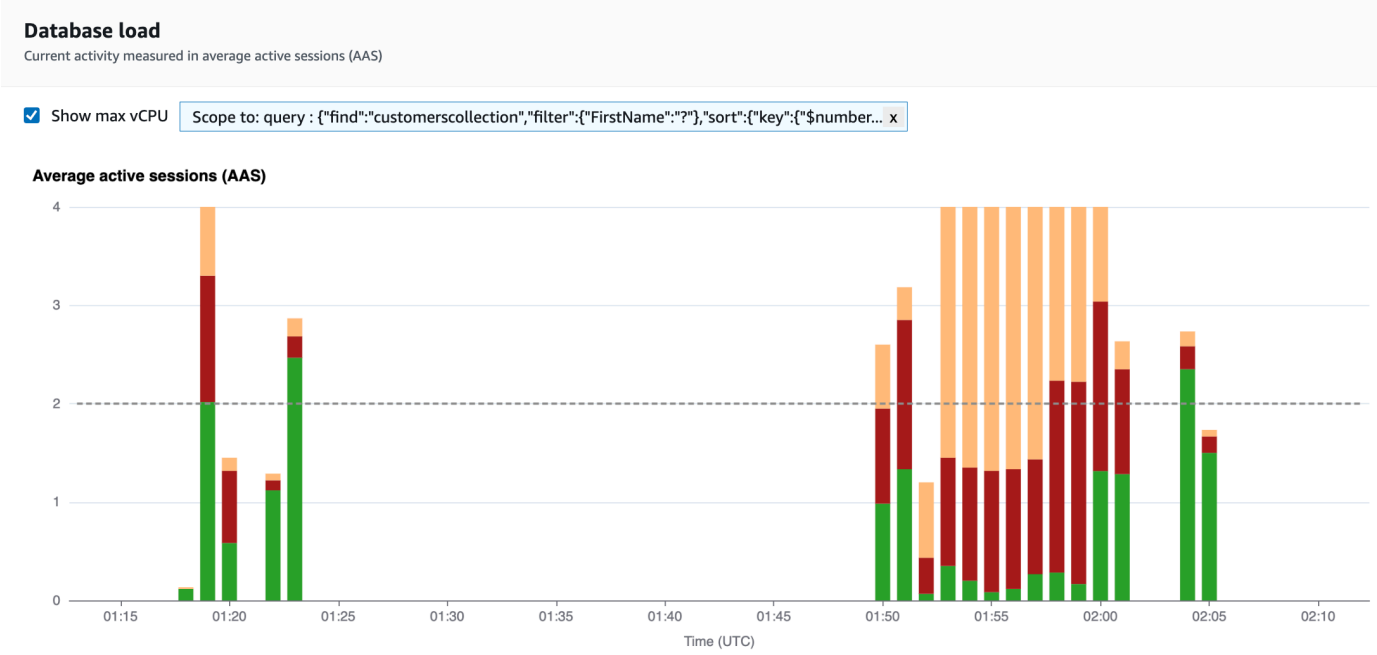
carico, la barra si riempie ed è di colore blu. Quando il carico supera il numero di CPU virtuali (vCPU) sulla classe di istanza, la barra diventa rossa, indicando un potenziale collo di bottiglia.



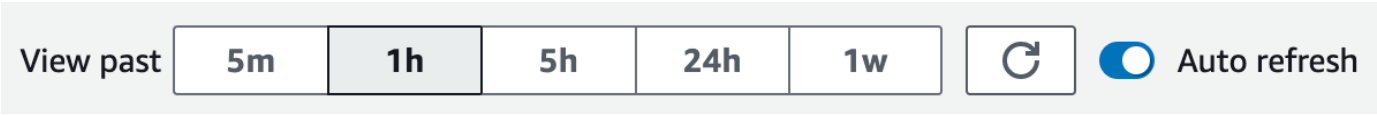
3. (Facoltativo) Scegliere un intervallo di tempo diverso selezionando un pulsante in alto a destra. Ad esempio, per modificare l'intervallo in 1 ora, seleziona 1h.



Nella schermata seguente, l'intervallo di caricamento del DB è di 1 ora.



4. Per aggiornare automaticamente i dati, abilita l'aggiornamento automatico.



Il pannello di controllo di Performance Insights si aggiorna automaticamente con i nuovi dati. La frequenza di aggiornamento dipende dalla quantità di dati visualizzati:

- Se scegli 5 minuti, l'aggiornamento avviene ogni 5 secondi.
- 1 ora si aggiorna ogni minuto.
- 5 ore si aggiorna ogni minuto.
- Se scegli 24 ore, l'aggiornamento avviene ogni 5 minuti.
- Se scegli 1 settimana, l'aggiornamento avviene ogni ora.

Analisi del carico del database in base agli stati di attesa

Se il grafico del carico del database (carico del database) mostra un collo di bottiglia, puoi scoprire da dove proviene il carico. A questo scopo, osserva la tabella Top Load Items (Elementi con carico) sotto la tabella Database load (Carico database). Scegli un elemento particolare, ad esempio una query o un'applicazione, per approfondirlo e visualizzarne i dettagli.

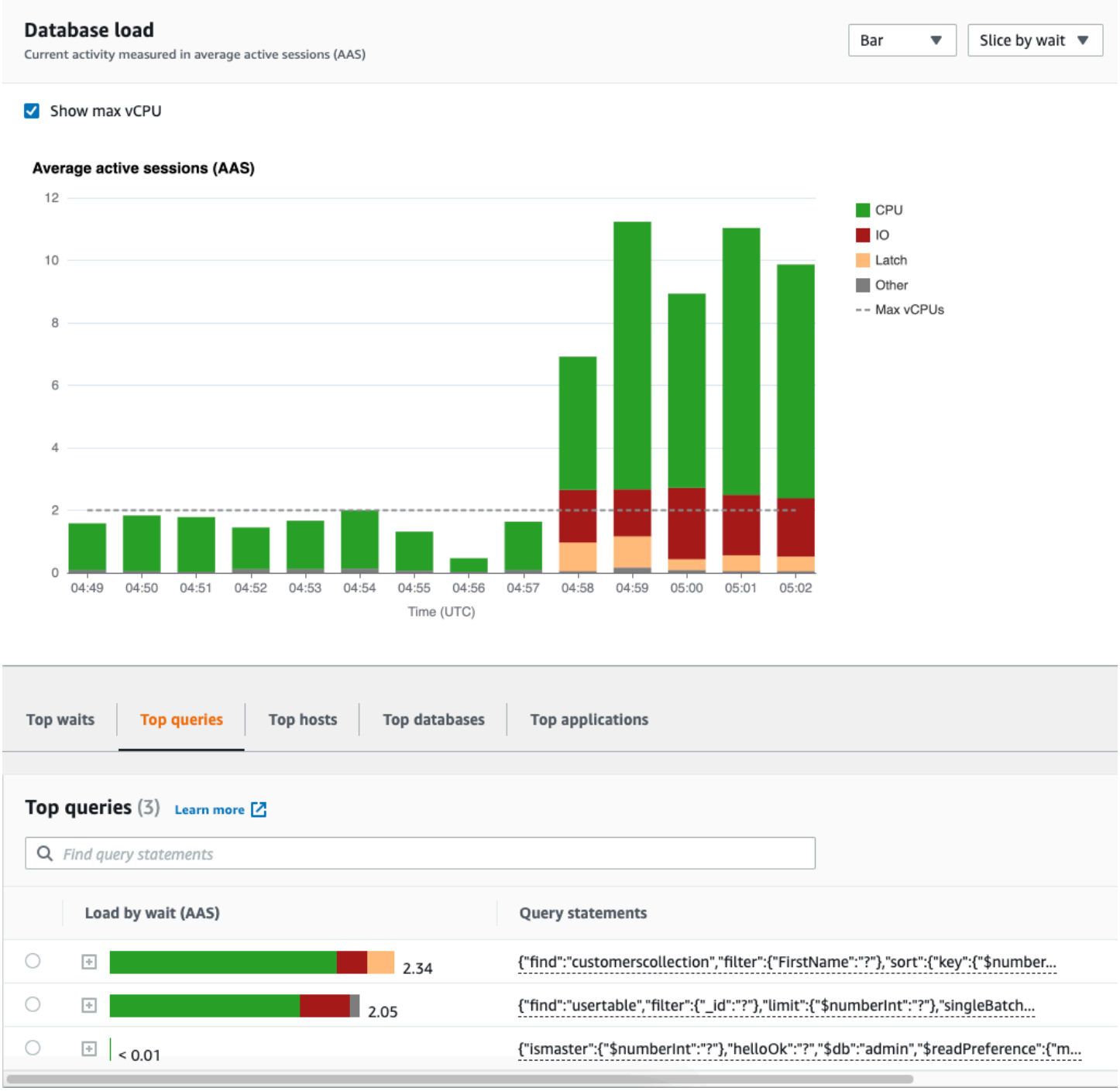
Il carico del database raggruppato per attese e principali query in genere fornisce la migliore comprensione dei problemi di prestazioni. Il carico del database raggruppato in base alle attese mostra la presenza di eventuali colli di bottiglia nel database relativamente alle risorse o alla simultaneità. In questo caso, la scheda Prime interrogazioni della tabella Elementi di caricamento principali mostra quali sono le query che generano quel carico.

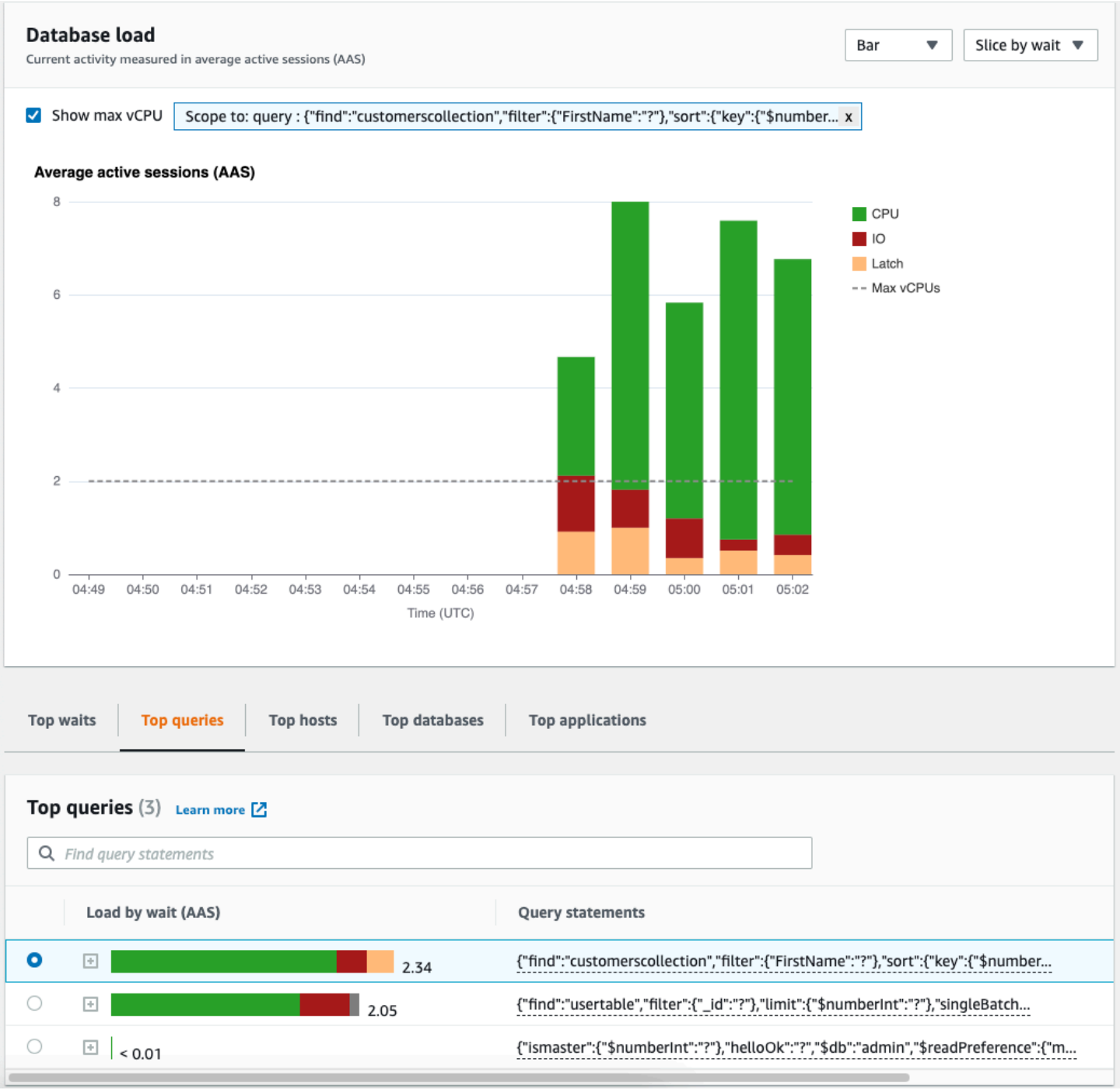
Il flusso di lavoro tipico per diagnosticare problemi di performance è il seguente:

1. Esaminare il grafico Database load (Carico database) per determinare se sono presenti eventi imprevisti di superamento della riga Max CPU (CPU max) da parte del carico del database.
2. Se sono presenti, osservare il grafico Database load (Carico database) e individuare lo stato o gli stati di attesa che sono i principali responsabili.
3. Identifica le query di riepilogo che causano il caricamento visualizzando quali delle query presenti nella scheda Top queries della tabella Top load items contribuiscono maggiormente a questi stati di attesa. Puoi identificarli dalla colonna Load by Wait (AAS).
4. Scegli una di queste interrogazioni riassuntive nella scheda Domande principali per espanderla e visualizzare le query secondarie da cui è composta.

Puoi anche vedere quali host o applicazioni stanno contribuendo maggiormente al carico selezionando rispettivamente I principali host o le principali applicazioni. I nomi delle applicazioni sono specificati nella stringa di connessione all'istanza Amazon DocumentDB. Unknownindica che il campo dell'applicazione non è stato specificato.

Ad esempio, nella dashboard seguente, la CPU attende l'account per la maggior parte del carico del DB. Selezionando la query principale in Principali interrogazioni, il grafico di carico del database si concentrerà sul carico maggiore apportato dalla query di selezione.





Panoramica della scheda Principali domande

Per impostazione predefinita, la scheda Query principale mostra le query che contribuiscono maggiormente al carico del DB. È possibile analizzare il testo della query per ottimizzare le query.

Argomenti

- Digest delle interrogazioni

- Caricamento per attesa (AAS)
- Visualizzazione di informazioni dettagliate sulle interrogazioni
- Accesso al testo della richiesta di dichiarazione
- Visualizzazione e download del testo della richiesta di dichiarazione

Digest delle interrogazioni

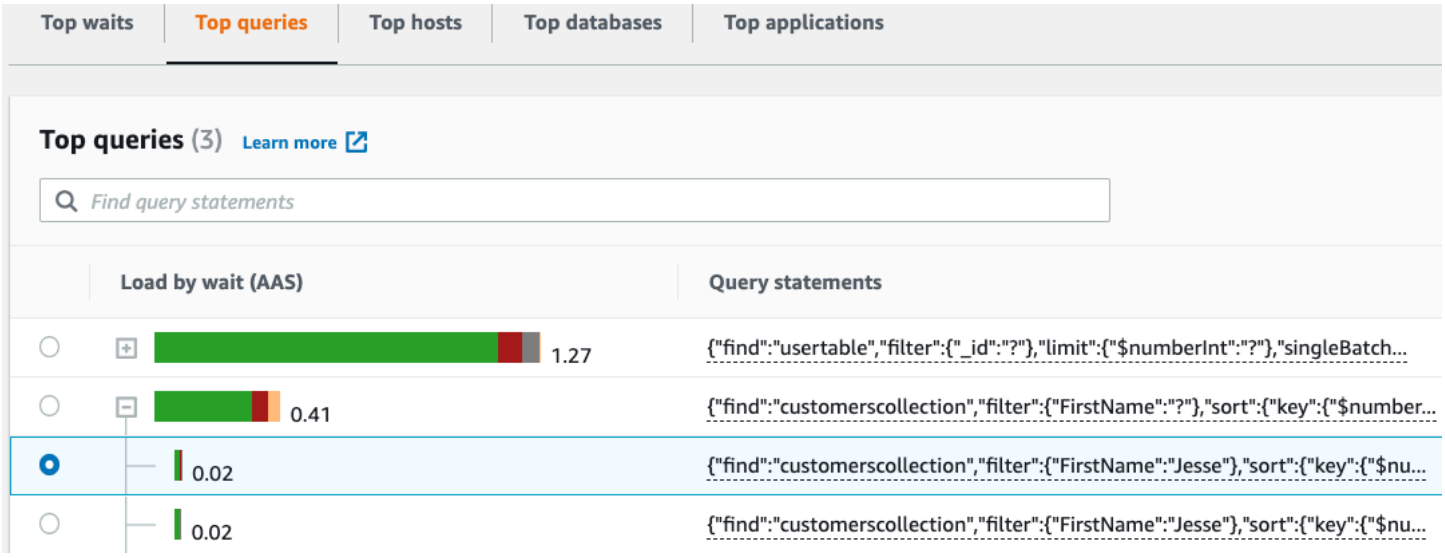
Un digest di query è un composto di più query effettive strutturalmente simili ma che potrebbero avere valori letterali diversi. Il digest sostituisce i valori codificati con un punto interrogativo. Ad esempio, un digest di query potrebbe avere il seguente aspetto:

```
{"find":"customerscollection","filter":{"FirstName":"?"},"sort":{"key":
{"$numberInt":"?"}},"limit":{"$numberInt":"?"}}
```

Questo digest può includere le seguenti query figlio:

```
{"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"}}
```

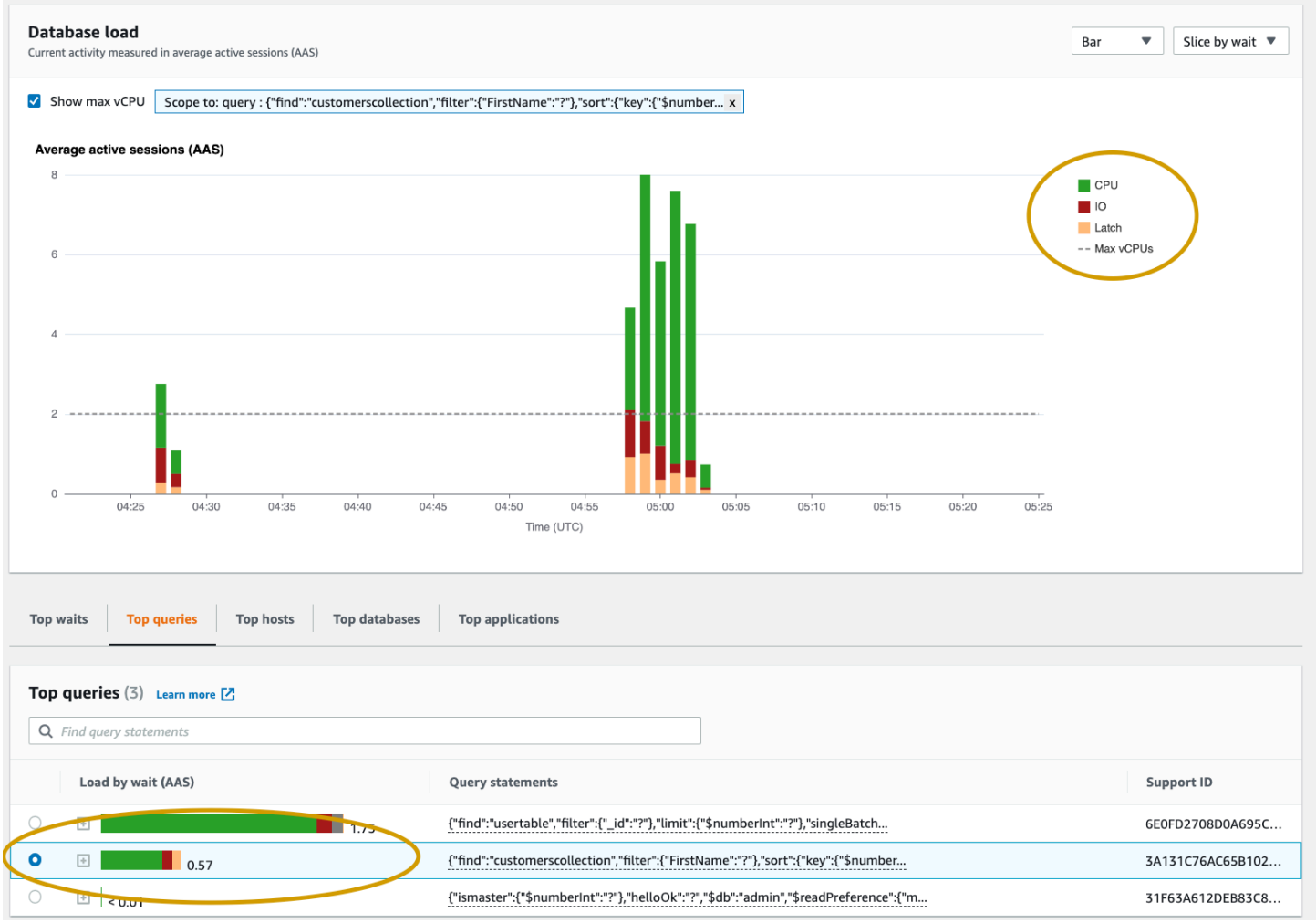
Per visualizzare le istruzioni di interrogazione letterali in un riassunto, selezionate la query, quindi scegliete il simbolo più (). + Nella schermata seguente, la query selezionata è un digest.



> **Note**
>
> Un query digest raggruppa istruzioni di interrogazione simili, ma non oscura le informazioni riservate.

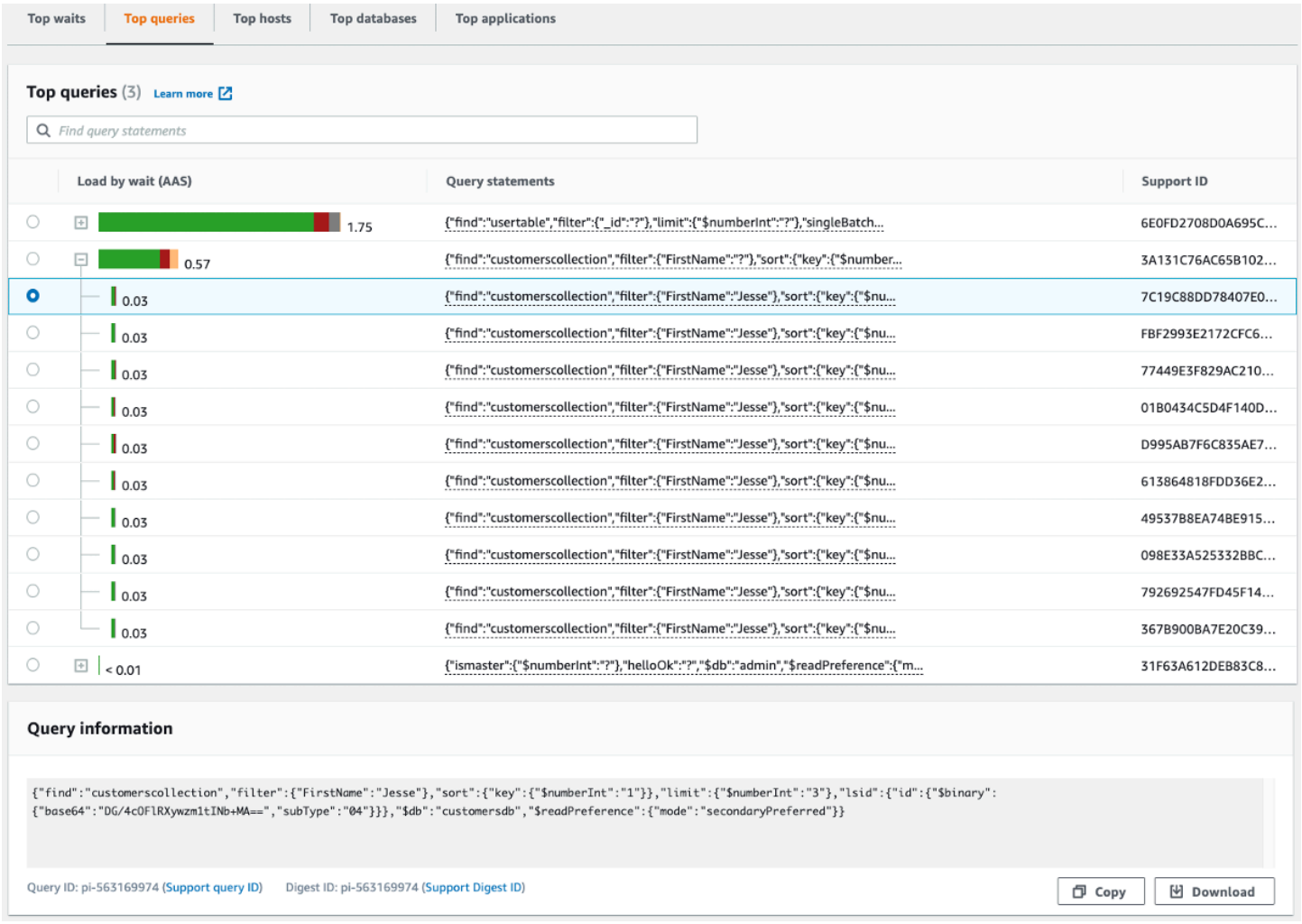
Caricamento per attesa (AAS)

Nelle principali query, la colonna Load by waits (AAS) illustra la percentuale del carico del database associata a ciascun elemento di caricamento principale. Questa colonna riflette il carico di quell'elemento in base al raggruppamento attualmente selezionato nel diagramma di carico del DB. Ad esempio, è possibile raggruppare il Carico DB in base agli stati di attesa. In questo caso, la barra DB Load by Waits (Carico del database in base alle attese) è dimensionata, segmentata e rappresentata da un colore per mostrare qual è il contributo della query a un dato stato di attesa. Mostra anche quali stati di attesa stanno influenzando la query selezionata.



Visualizzazione di informazioni dettagliate sulle interrogazioni

Nella tabella Top query, è possibile aprire un'istruzione riepilogativa per visualizzarne le informazioni. Le informazioni vengono visualizzate nel riquadro inferiore.



I seguenti tipi di identificatori (ID) sono associati alle istruzioni di interrogazione:

1. ID della query di supporto: un valore hash dell'ID della query. Questo valore serve solo per fare riferimento a un ID di query quando si lavora con AWS il supporto. AWS L'assistenza non ha accesso agli ID e al testo della query effettivi.
2. Support digest ID: un valore hash dell'ID del digest. Questo valore serve solo per fare riferimento a un ID digest quando si lavora con Support. AWS AWS L'assistenza non ha accesso agli ID di riepilogo effettivi e al testo della richiesta.

Accesso al testo della richiesta di dichiarazione

Per impostazione predefinita, ogni riga della tabella Prime domande mostra 500 byte di testo di interrogazione per ogni istruzione di interrogazione. Quando un'istruzione digest supera i 500 byte, è possibile visualizzare altro testo aprendo l'istruzione nella dashboard di Performance Insights. In questo caso, la lunghezza massima per l'interrogazione visualizzata è 1 KB. Se visualizzi una dichiarazione di interrogazione completa, puoi anche scegliere Scarica.

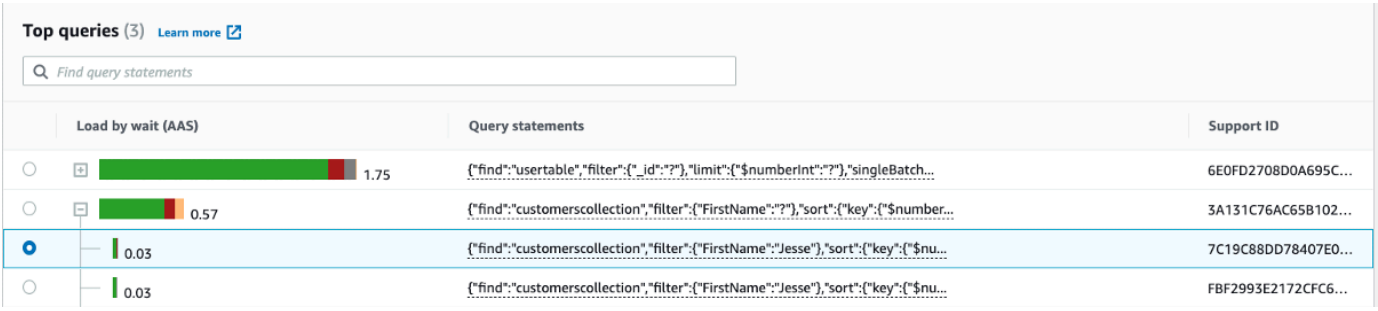
Visualizzazione e download del testo della richiesta di dichiarazione

Nella dashboard di Performance Insights, puoi visualizzare o scaricare il testo della query.

Per visualizzare altro testo della query nella dashboard di Performance Insights

1. Apri la console Amazon DocumentDB all'indirizzo: https://console.aws.amazon.com/docdb/
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli istanza database. Viene visualizzato il pannello di controllo di Performance Insights per l'istanza database.

   Le istruzioni di interrogazione con testo più grande di 500 byte avranno l'aspetto seguente:



4. Esamina la sezione delle informazioni sulla query per visualizzare una parte maggiore del testo dell'interrogazione.



La dashboard di Performance Insights può visualizzare fino a 1 KB per ogni istruzione di interrogazione completa.

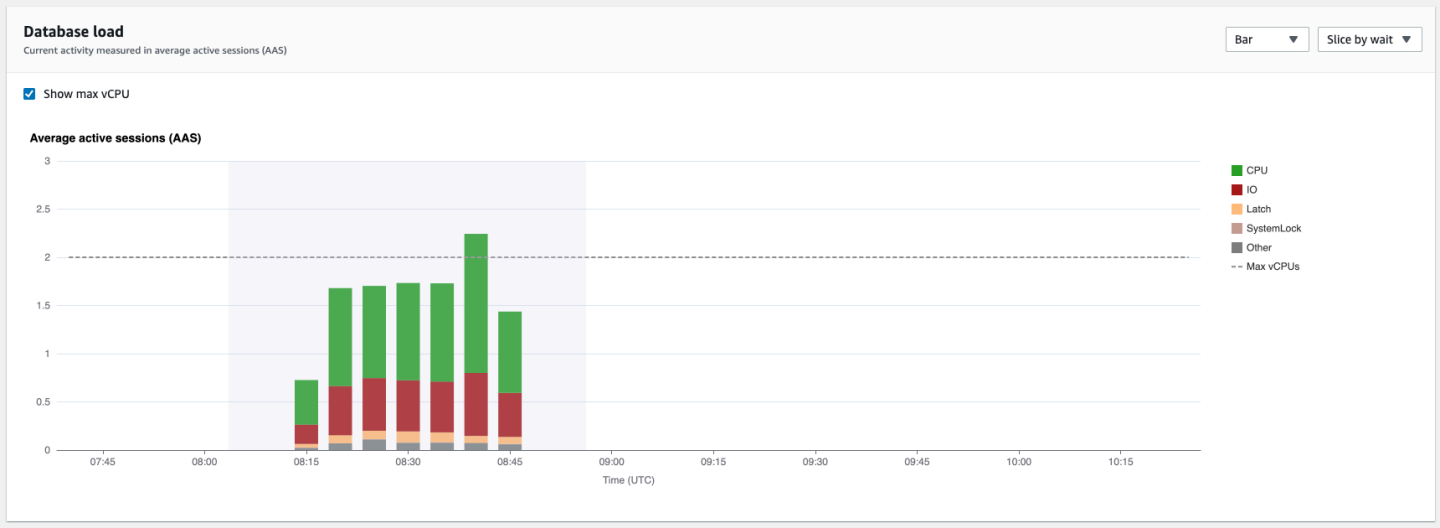
> **Note**
>
> Per copiare o scaricare l'istruzione di interrogazione, disabilita eventuali blocchi popup.

Ingrandimento del diagramma di carico del database

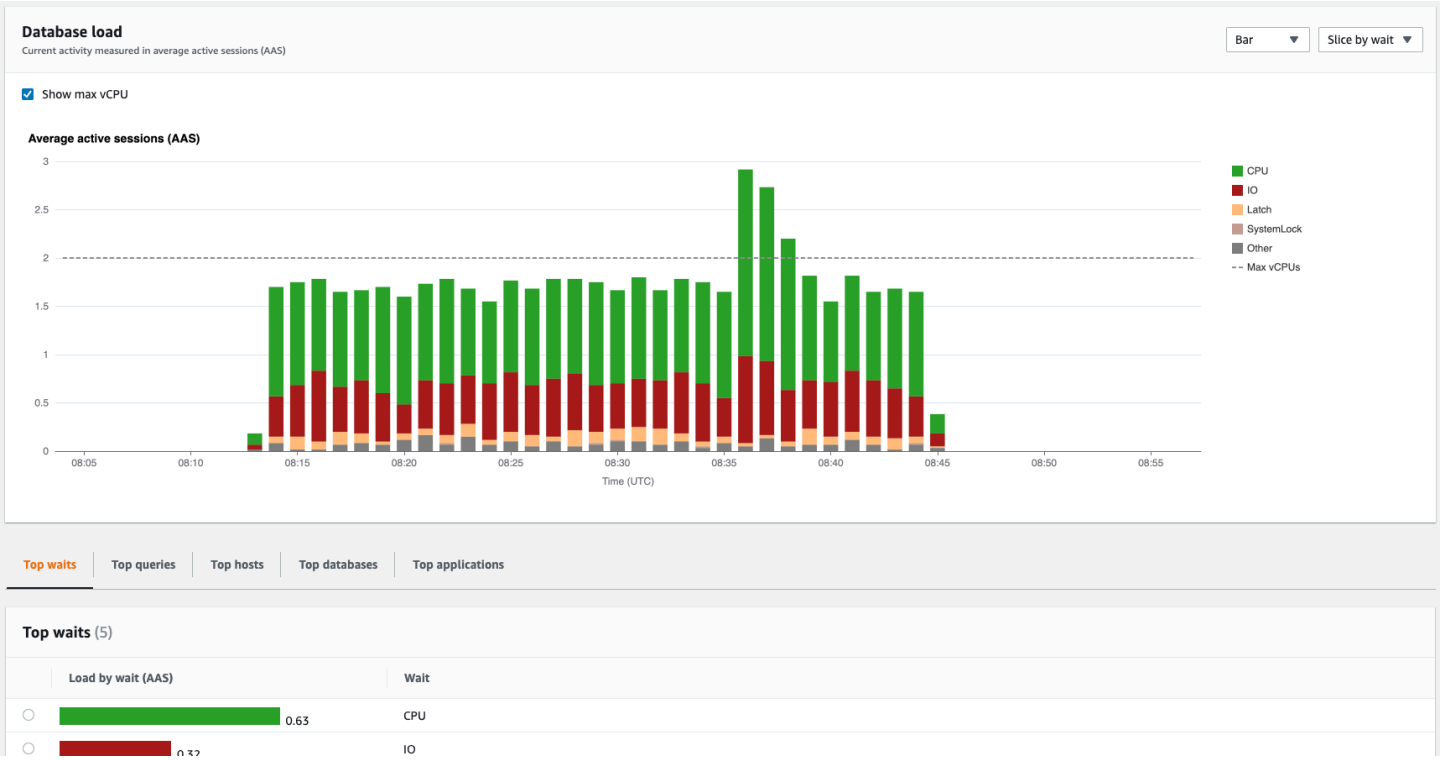
Si possono utilizzare altre funzionalità dell'interfaccia utente di Performance Insights per analizzare i dati sulle performance.

Click-and-Drag Ingrandisci

Nell'interfaccia di Performance Insights, puoi scegliere una piccola parte del grafico di carico e ingrandire il dettaglio.



Per ingrandire una parte del grafico di carico, scegli l'ora di inizio e trascina fino alla fine del periodo di tempo che ti interessa. Una volta fatto, l'area selezionata è evidenziata. Quando rilasciate il mouse, il diagramma di carico ingrandisce l'area selezionata e la tabella Primi elementi viene ricalcolata.



Recupero dei parametri con l'API Performance Insights

Quando Performance Insights è abilitato, l'API fornisce visibilità sulle prestazioni dell'istanza. Amazon CloudWatch Logs fornisce la fonte autorevole per le metriche di monitoraggio vendute per i servizi. AWS

Performance Insights offre una vista specifica del dominio del carico del database misurato come numero medio di sessioni attive (AAS). Questo parametro viene visualizzata dai consumer API come un set di dati temporali bidimensionali. La dimensione temporale dei dati fornisce i dati relativi al carico del database per ogni momento dell'intervallo di tempo in cui è stata eseguita la query. Ogni punto temporale scompone il carico complessivo in relazione alle dimensioni richieste, come `Query`, `Wait-state`, `Application` o `Host`, misurato in corrispondenza di quel punto temporale.

Amazon DocumentDB Performance Insights monitora l'istanza DB di Amazon DocumentDB in modo da poter analizzare e risolvere i problemi relativi alle prestazioni del database. Un modo per visualizzare i dati di Performance Insights è disponibile nella Console di gestione AWS. Performance Insights fornisce inoltre un'API pubblica per eseguire query sui dati. Puoi usare l'API per effettuare quanto segue:

- Scaricamento dei dati in un database
- Aggiungi dati Performance Insights ai pannelli di controllo di monitoraggio esistenti

- Crea strumenti di monitoraggio

Per utilizzare l'API Performance Insights, abilita Performance Insights su una delle tue istanze Amazon DocumentDB. Per informazioni sull'abilitazione di Performance Insights, consulta Abilitazione e disattivazione di Performance Insights. Per ulteriori informazioni sull'API di Performance Insights, consulta la Documentazione di riferimento dell'API di Performance Insights.

L'API di Performance Insights fornisce le seguenti operazioni.

Operazione di Performance Insights	AWS CLI comando	Description
`DescribeDimensionKeys`	`aws pi describe-dimension-keys`	Recupera le prime N chiavi di dimensione per un parametro per un determinato periodo di tempo.
`GetDimensionKeyDetails`	`aws pi get-dimension-key-details`	Recupera gli attributi del gruppo di dimensioni specificato per un'istanza database o un'origine dati. Ad esempio, se si specifica un ID di query e se i dettagli della dimensione sono disponibili, `GetDimensionKeyDetails` recupera il testo completo della dimensione `db.query.statement` associata a questo ID. Questa operazione è utile perché `GetResourceMetrics` e `DescribeDimensionKeys` non supporta il recupero di testo di istruzioni di interrogazione di grandi dimensioni.

Operazione di Performance Insights	AWS CLI comando	Description
GetResourceMetadata	aws pi get-resource-metadata	Recupera i metadati per diverse caratteristiche. Ad esempio, i metadati potrebbero indicare che una caratteristica è attivata o disattivata su un'istanza database specifica.
GetResourceMetrics	aws pi get-resource-metrics	Recupera parametri Performance Insights per un set di origini dati, su un periodo di tempo. Puoi fornire gruppi di dimensioni e dimensioni specifiche e fornire criteri di aggregazione e filtro per ogni gruppo.
ListAvailableResourceDimensions	aws pi list-available-resource-dimensions	Recupera le dimensioni su cui è possibile eseguire query per ogni tipo di parametro specificato su un'istanza specificata.
ListAvailableResourceMetrics	aws pi list-available-resource-metrics	Recupera tutti i parametri disponibili dei tipi di parametro specificati su cui è possibile eseguire query per un'istanza database specificata.

Argomenti

- AWS CLI per Performance Insights
- Recupero dei parametri di serie temporali
- AWS CLI esempi di Performance Insights

AWS CLI per Performance Insights

Puoi visualizzare i dati di Performance Insights utilizzando la AWS CLI. Puoi visualizzare la guida per i comandi AWS CLI per Performance Insights inserendo quanto segue nella riga di comando.

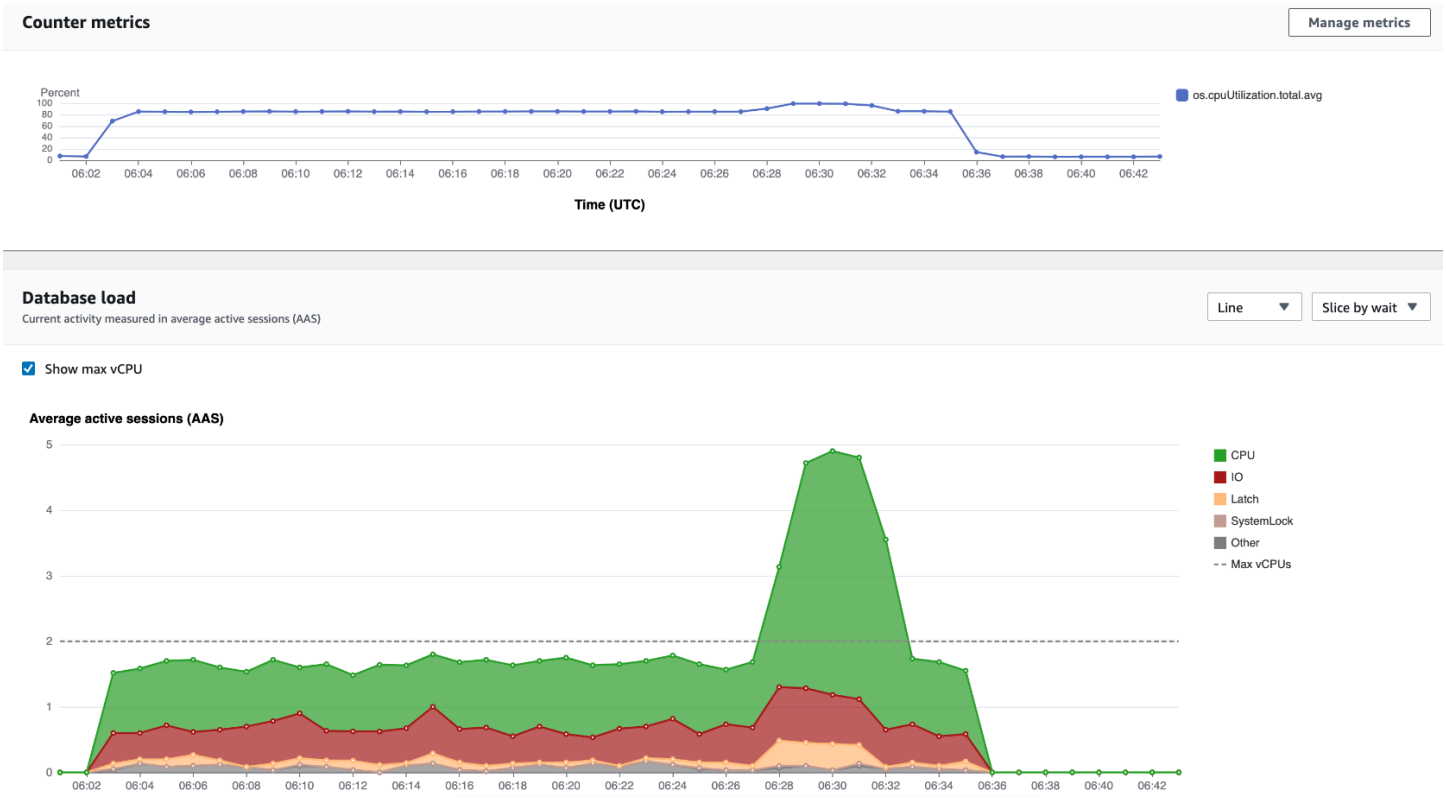
```
aws pi help
```

Se non hai AWS CLI installato l'interfaccia a riga di comando, vedi Installazione dell'interfaccia a riga di AWS comando nella Guida per AWS CLI l'utente per informazioni sull'installazione.

Recupero dei parametri di serie temporali

L'operazione `GetResourceMetrics` recupera uno o più parametri di serie temporali dai dati di Performance Insights. `GetResourceMetrics` richiede un parametro e un periodo di tempo e restituisce una risposta con un elenco di punti di dati.

Ad esempio, gli Console di gestione AWS usi `GetResourceMetrics` per compilare il grafico Counter Metrics e il diagramma Database Load, come mostrato nell'immagine seguente.



Tutti i parametri restituiti da `GetResourceMetrics` sono parametri di serie temporali standard ad eccezione di `db.load`. Questo parametro è visualizzato nel grafico Database Load (Carico del database). Il parametro `db.load` è diverso dagli altri parametri di serie temporali in quanto

può essere suddiviso in sottocomponenti detti dimensioni. Nell'immagine precedente, `db.load` è suddiviso e raggruppato in base agli stati delle attese che formano il `db.load`.

> **Note**
>
> `GetResourceMetrics` può anche restituire il parametro `db.sampleload`, ma il parametro `db.load` è appropriato nella maggior parte dei casi.

Per informazioni sui parametri contatore restituiti da `GetResourceMetrics`, consulta Performance Insights per le contrometriche.

I seguenti calcoli sono supportati per i parametri:

- Media: il valore medio per il parametro su un periodo di tempo. Aggiungi `.avg` al nome parametro.
- Minimo: il valore minimo per il parametro su un periodo di tempo. Aggiungi `.min` al nome parametro.
- Massimo: il valore massimo per il parametro su un periodo di tempo. Aggiungi `.max` al nome parametro.
- Somma: la somma dei valori dei parametri su un periodo di tempo. Aggiungi `.sum` al nome parametro.
- Conteggio di esempio: il numero di volte che il parametro è stato raccolto su un periodo di tempo. Aggiungi `.sample_count` al nome parametro.

Ad esempio, supponiamo che un parametro venga raccolto per 300 secondi (5 minuti) e che il parametro venga raccolto una volta al minuto. I valori per ogni minuto sono 1, 2, 3, 4 e 5. In questo caso. vengono restituiti i seguenti calcoli:

- Media: 3
- Minimo: 1
- Massimo: 5
- Somma: 15
- Conteggio del campione: 5

Per informazioni sull'utilizzo del `get-resource-metrics` AWS CLI comando, vedere. `get-resource-metrics`

Per l'opzione `--metric-queries`, specifica una o più query per cui ottenere risultati. Ciascuna query consiste di un parametro obbligatorio `Metric` e parametri facoltativi `GroupBy` e `Filter`. Di seguito è riportato un esempio della specifica di un'opzione `--metric-queries`.

```
{
   "Metric": "string",
   "GroupBy": {
     "Group": "string",
     "Dimensions": ["string", ...],
     "Limit": integer
   },
   "Filter": {"string": "string"
     ...}
```

AWS CLI esempi di Performance Insights

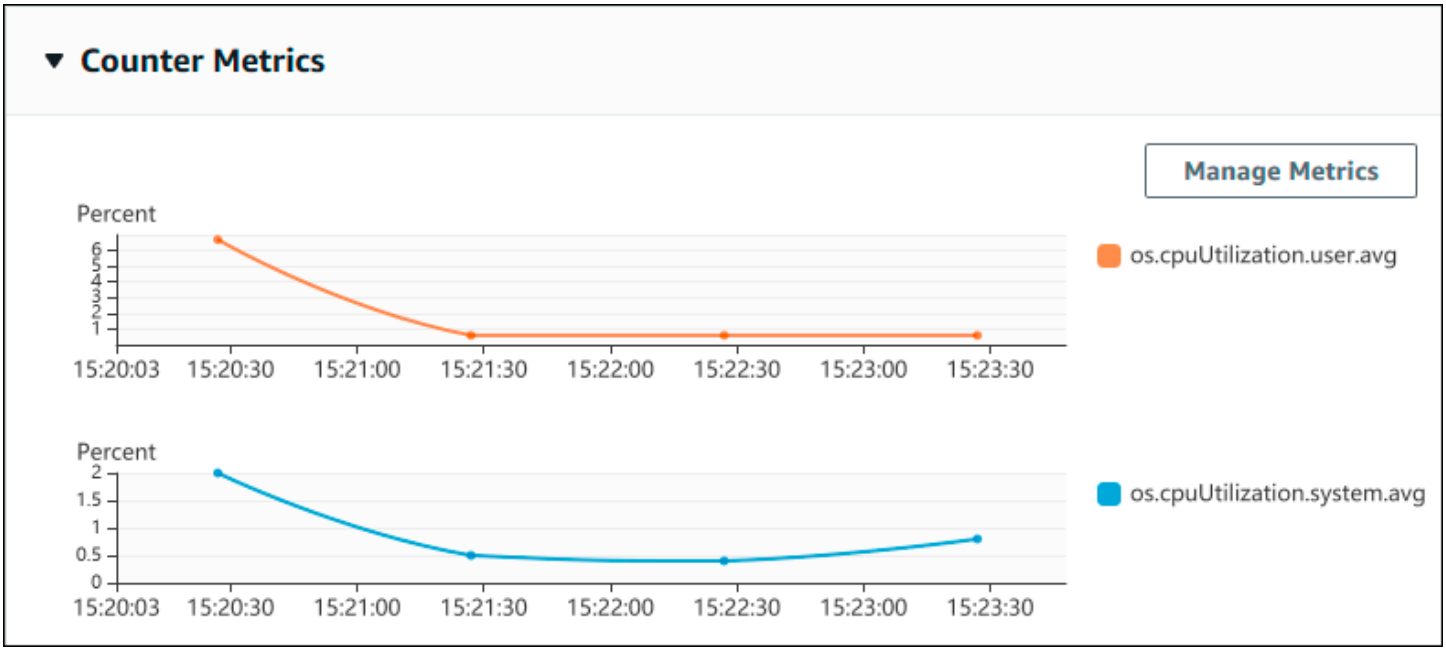
Gli esempi seguenti mostrano come utilizzare il AWS CLI for Performance Insights.

Argomenti

- Recupero dei parametri contatore
- Recupero della media del carico del DB per i principali stati di attesa
- Recupero del carico medio del DB per Top Query
- Recupero del carico medio del database filtrato per Query

Recupero dei parametri contatore

Lo screenshot seguente mostra due grafici dei parametri contatore nella Console di gestione AWS.



L'esempio seguente mostra come raccogliere gli stessi dati che utilizza la Console di gestione AWS per generare i due grafici dei parametri contatore.

Per Linux, macOS o 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"}]'
```

Per 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"}]'
```

Puoi agevolare la lettura del comando specificando un file per l'opzione `--metrics-query`. Il seguente esempio utilizza un file denominato query.json per l'opzione. Il file presenta i seguenti contenuti.

```
[
    {
        "Metric": "os.cpuUtilization.user.avg"
    },
    {
        "Metric": "os.cpuUtilization.idle.avg"
    }
]
```

Esegui il comando seguente per utilizzare il file.

Per Linux, macOS o 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
```

Per 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
```

L'esempio precedente specifica i seguenti valori per le opzioni:

- `--service-type`— DOCDB per Amazon DocumentDB
- `--identifier` – L'ID risorsa per l'istanza database
- `--start-time` e `--end-time` – I valori ISO 8601 `DateTime` per il periodo su cui eseguire le query, con supporto di più formati

Esegue query per un intervallo di tempo di un'ora:

- `--period-in-seconds` – `60` per una query al minuto
- `--metric-queries` – Una serie di due query, ognuna solo per un parametro

  Il nome del parametro utilizza punti per classificare il parametro in una categoria utile, dove l'ultimo elemento è una funzione. Nell'esempio, la funzione è `avg` per ciascuna query. Come per Amazon CloudWatch, le funzioni supportate sono`min`, `maxtotal`, `eavg`.

La risposta è simile a quella riportata di seguito.

```
{
    "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
```

La risposta presenta un `Identifier`, un `AlignedStartTime` e un `AlignedEndTime`. Poiché il valore `--period-in-seconds` era `60`, l'ora di inizio e fine è stata allineata al minuto. Se `--period-in-seconds` fosse stato `3600`, l'ora di inizio e fine sarebbe stata allineata all'ora.

`MetricList` nella risposta ha una serie di voci, ciascuna con una voce `Key` e una voce `DataPoints`. Ciascun `DataPoint` ha un `Timestamp` e un `Value`. Ciascun elenco `Datapoints` ha 60 punti di dati in quanto le query sono per dati al minuto nell'arco di un'ora, con `Timestamp1`/`Minute1`, `Timestamp2`/`Minute2` e così via, fino a `Timestamp60`/`Minute60`.

Poiché la query è per due diversi parametri contatore, la risposta contiene due element `MetricList`.

Recupero della media del carico del DB per i principali stati di attesa

L'esempio seguente è la stessa query Console di gestione AWS utilizzata per generare un grafico a linee ad area in pila. Questo esempio recupera il valore dell'`db.load.avg`ultima ora con il carico diviso in base ai primi sette stati di attesa. Il comando è come quello in Recupero dei parametri contatore. Tuttavia, il file query.json presenta i seguenti contenuti.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_state", "Limit": 7 }
    }
]
```

Eseguire il comando riportato qui di seguito.

Per Linux, macOS o 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
```

Per 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
```

L'esempio specifica la metrica `db.load.avg` e uno dei primi sette `GroupBy` stati di attesa. Per informazioni dettagliate sui valori validi per questo esempio, consulta DimensionGroup la sezione Performance Insights API Reference.

La risposta è simile a quella riportata di seguito.

```
{
    "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 questa risposta, ci sono tre voci nel`MetricList`. C'è una voce per il totale `db.load.avg` e tre voci ciascuna per il `db.load.avg` diviso in base a uno dei primi tre stati di attesa. Poiché esisteva una dimensione di raggruppamento (a differenza del primo esempio), deve essere presente una chiave per ogni raggruppamento della metrica. Può esserci una sola chiave per ciascun parametro, come nel caso d'uso del parametro contatore di base.

Recupero del carico medio del DB per Top Query

L'esempio seguente raggruppa `db.wait_state` le prime 10 istruzioni di interrogazione. Esistono due diversi gruppi per le istruzioni di interrogazione:

- `db.query`— L'istruzione di interrogazione completa, ad esempio `{"find":"customers","filter":{"FirstName":"Jesse"},"sort":{"key": {"$numberInt":"1"}}}`
- `db.query_tokenized`— L'istruzione di interrogazione tokenizzata, ad esempio `{"find":"customers","filter":{"FirstName":"?"},"sort":{"key": {"$numberInt":"?"}},"limit":{"$numberInt":"?"}}`

Quando si analizzano le prestazioni del database, può essere utile considerare le istruzioni di interrogazione che differiscono solo per i parametri come un unico elemento logico. Pertanto, puoi utilizzare `db.query_tokenized` durante le query. Tuttavia, soprattutto quando si è interessati`explain()`, a volte è più utile esaminare le istruzioni di interrogazione complete con parametri. Esiste una relazione padre-figlio tra le query tokenizzate e quelle complete, con più query complete (figli) raggruppate nella stessa query tokenizzata (padre).

Il comando in questo esempio è simile a quello in Recupero della media del carico del DB per i principali stati di attesa. Tuttavia, il file query.json presenta i seguenti contenuti.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.query_tokenized", "Limit": 10 }
    }
]
```

Nell'esempio seguente viene utilizzato `db.query_tokenized`.

Per Linux, macOS o 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
```

Per 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
```

Questo esempio esegue query della durata di 1 ora, con un intervallo di un minuto in secondi.

L'esempio specifica la metrica `db.load.avg` e uno dei sette stati `GroupBy` di attesa principali. Per informazioni dettagliate sui valori validi per questo esempio, consulta `DimensionGroup` la sezione Performance Insights API Reference.

La risposta è simile a quella riportata di seguito.

```
{
    "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
```

Questa risposta contiene 11 voci `MetricList` (1 totale, 10 query tokenizzate principali), ognuna delle quali ha 24 all'ora. `DataPoints`

Per le query tokenizzate, ci sono tre voci in ogni elenco di dimensioni:

- `db.query_tokenized.statement`— L'istruzione di interrogazione tokenizzata.
- `db.query_tokenized.db_id` — L'ID sintetico che Performance Insights genera per te. Questo esempio restituisce l'ID sintetico `pi-1064184600`.
- `db.query_tokenized.id` – L'ID della query all'interno di Performance Insights.

  In Console di gestione AWS, questo ID è denominato Support ID. Si chiama così perché l'ID è costituito da dati che il AWS supporto può esaminare per aiutarti a risolvere un problema con il tuo database. AWS prende molto sul serio la sicurezza e la privacy dei tuoi dati e quasi tutti i dati vengono archiviati crittografati con il tuo. AWS KMS key Pertanto, nessuno all'interno AWS può esaminare questi dati. Nell'esempio precedente, sia `tokenized.statement` che `tokenized.db_id` vengono archiviati crittografati. In caso di problemi con il database, l' AWS assistenza può aiutarti facendo riferimento all'ID di supporto.

Quando si eseguo query, potrebbe essere utile specificare un `Group` in `GroupBy`. Tuttavia, per un controllo più dettagliato dei dati restituiti, occorre specificare l'elenco delle dimensioni. Ad esempio, se tutto ciò di cui si necessita è `db.query_tokenized.statement`, è possibile aggiungere l'attributo `Dimensions` al file query.json.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": {
            "Group": "db.query_tokenized",
            "Dimensions":["db.query_tokenized.statement"],
            "Limit": 10
        }
    }
]
```

Recupero del carico medio del database filtrato per Query

La query dell'API corrispondente in questo esempio è simile al comando in Recupero del carico medio del DB per Top Query. Tuttavia, il file query.json presenta i seguenti contenuti.

```
[
 {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_state", "Limit": 5  },
        "Filter": { "db.query_tokenized.id": "AKIAIOSFODNN7EXAMPLE" }
```

```
    }
]
```

In questa risposta, tutti i valori vengono filtrati in base al contributo della query tokenizzata AKIAIOSFODNN7EXAMPLE specificata nel file query.json. Le chiavi potrebbero inoltre seguire un ordine diverso rispetto a una query senza filtro, poiché sono i cinque principali stati di attesa a influire sulla query filtrata.

CloudWatch Metriche di Amazon per Performance Insights

Performance Insights pubblica automaticamente le metriche su Amazon. CloudWatch Gli stessi dati possono essere interrogati da Performance Insights, ma la presenza delle metriche CloudWatch semplifica l'aggiunta di allarmi. CloudWatch Agevola inoltre l'aggiunta dei parametri ai pannelli di controllo di CloudWatch esistenti.

Metrica	Descrizione
DBLoad	Il numero di sessioni attive per Amazon DocumentDB. Generalmente, si richiedono i dati per il numero medio di sessioni attive. In Performance Insights, questi dati sono oggetto di query come `db.load.avg` .
DBLoadCPU	Il numero di sessioni attive in cui il tipo di stato di attesa è CPU. In Performance Insights, questi dati vengono interrogati come`db.load.avg` , filtrati in base al tipo di stato di attesa. `CPU`
DBLoadNonCPU	Il numero di sessioni attive in cui il tipo di stato di attesa non è CPU.

> **Note**
>
> Queste metriche vengono pubblicate CloudWatch solo in caso di carico sull'istanza DB.

Puoi esaminare queste metriche utilizzando la CloudWatch console, l'API o l' AWS CLI CloudWatchAPI.

Ad esempio, puoi ottenere le statistiche per il parametro `DBLoad` eseguendo il comando get-metric-statistics.

```
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
```

Questo esempio genera un output simile a quello riportato di seguito.

```
{
    "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"
}
```

Puoi utilizzare la funzione matematica delle `DB_PERF_INSIGHTS` metriche nella CloudWatch console per interrogare le metriche dei contatori di Amazon DocumentDB Performance Insights. La `DB_PERF_INSIGHTS` funzione include anche la `DBLoad` metrica a intervalli inferiori ai minuti. Puoi impostare CloudWatch allarmi su queste metriche. Per maggiori dettagli su come creare un allarme, consulta Creare un avviso sulle metriche dei contatori di Performance Insights da un database. AWS

Per ulteriori informazioni CloudWatch, consulta Cos'è Amazon CloudWatch? nella Guida per CloudWatch l'utente di Amazon.

Performance Insights per le contrometriche

Le contrometriche sono metriche del sistema operativo nella dashboard di Performance Insights. Per agevolare l'individuazione e l'analisi di problemi legati alle prestazioni, è possibile correlare i parametri contatore ai carichi dei database.

Contatori del sistema operativo in Performance Insights

I seguenti contatori del sistema operativo sono disponibili con Amazon DocumentDB Performance Insights.

Contatore	Tipo	Parametro
attive	memory	os.memory.active
buffers	memory	os.memory.buffers
cached	memory	os.memory.cached
dirty	memory	os.memory.dirty

Contatore	Tipo	Parametro
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
user	cpuUtilization	os.cpu Utilization.user
wait	cpuUtilization	os.cpu Utilization.wait
one	caricare AverageMinute	os.load AverageMinute.one
fifteen	caricare AverageMinute	os.load AverageMinute.fifteen
cinque	caricare 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

Contatore	Tipo	Parametro
rx	network	os.network.rx
tx	network	os.network.tx
numVCPUs	general	os.general.numVCPUs

Utilizzo di Amazon DocumentDB serverless

Amazon DocumentDB serverless è una configurazione on-demand con scalabilità automatica che automatizza i processi di regolazione dinamica della capacità del database Amazon DocumentDB in base alla richiesta dell'applicazione. Ti verranno addebitate solo le risorse consumate dai cluster. Pertanto, DocumentDB serverless può aiutarti a rispettare il budget ed evitare di pagare per risorse informatiche che non utilizzi.

Questo tipo di automazione è particolarmente utile per ambienti con carichi di lavoro altamente variabili e imprevedibili, come database multitenant, database distribuiti e sistemi di sviluppo e test.

Argomenti

- Casi d'uso serverless di DocumentDB
- Vantaggi di Amazon DocumentDB serverless
- Come funziona Amazon DocumentDB serverless
- Requisiti e limitazioni per DocumentDB serverless
- Creazione di un cluster che utilizza Amazon DocumentDB serverless
- Migrazione ad Amazon DocumentDB serverless
- Gestione di Amazon DocumentDB senza server
- Limiti delle istanze serverless di Amazon DocumentDB
- Configurazione di scalabilità serverless di Amazon DocumentDB
- Monitoraggio di Amazon DocumentDB senza server

Casi d'uso serverless di DocumentDB

Sia i cluster con provisioning di Amazon DocumentDB che quelli serverless di Amazon DocumentDB supportano molti tipi di carichi di lavoro di database, dagli ambienti di sviluppo e test alle applicazioni aziendali più impegnative e critiche che richiedono scalabilità e disponibilità elevate. Ma DocumentDB serverless aggiunge un'altra dimensione ai carichi di lavoro dei clienti, vale a dire la capacità di supportare siti Web e applicazioni con carichi di lavoro imprevedibili.

DocumentDB serverless è particolarmente utile per i seguenti casi d'uso:

- Carichi di lavoro variabili: stai eseguendo carichi di lavoro con aumenti di attività improvvisi e imprevedibili. Un esempio può essere quello di un sito che offre informazioni sulla viabilità, che

riscontra un forte aumento dell'attività quando inizia a piovere. Un altro è un sito di e-commerce che aumenta il proprio traffico durante un'offerta o una vendita promozionale. Con DocumentDB serverless, il database ridimensiona automaticamente la capacità per soddisfare le esigenze di picco di carico dell'applicazione e si ridimensiona quando l'aumento di attività è terminato. Con DocumentDB serverless, non è più necessario effettuare il provisioning della capacità media o di picco. È possibile specificare un limite massimo di capacità per gestire la peggiore situazione possibile, che viene raggiunto solo in caso di necessità.

  - La granularità della scalabilità in DocumentDB serverless consente di adattare la capacità in modo strettamente adeguato alle esigenze del database. Per un cluster con provisioning, la scalabilità richiede l'aggiunta di un'istanza completamente nuova. DocumentDB serverless può aggiungere mezza DCU quando è necessaria solo un po' più di capacità. Può aggiungere 0,5, 1, 1,5, 2 o mezze DCU aggiuntive in base alla capacità aggiuntiva necessaria per gestire un aumento del carico di lavoro. Inoltre, può rimuovere 0,5, 1, 1,5, 2 o altre mezze DCU quando il carico di lavoro diminuisce e tale capacità non è più necessaria.

- Multi-tenant applicazioni: con DocumentDB serverless, non è necessario gestire individualmente la capacità del database per ciascuna applicazione del parco applicazioni. DocumentDB serverless gestisce la capacità dei singoli database per te.
  - È possibile creare un cluster per ciascun tenant. In questo modo, è possibile utilizzare funzionalità come la clonazione e il ripristino delle istantanee per migliorare l'alta disponibilità e il disaster recovery in base alle esigenze di ciascun tenant.
  - Ogni tenant potrebbe avere periodi di attività e inattività specifici, che variano a seconda dell'ora, del periodo dell'anno, degli eventi promozionali e così via. Ogni cluster è dotato di un'ampia gamma di capacità. In questo modo, i cluster con bassa attività comportano costi di istanza minimi. Tutti i cluster sono in grado di scalare rapidamente per gestire periodi di attività elevata.
- Nuove applicazioni: stai implementando una nuova applicazione e non sei sicuro delle dimensioni dell'istanza di cui hai bisogno. Utilizzando DocumentDB serverless, puoi configurare un cluster con una o più istanze e fare in modo che il database si scali automaticamente in base ai requisiti di capacità dell'applicazione.
- Mixed-use applicazioni: supponiamo di avere un'applicazione OLTP (Online Transaction Processing), ma periodicamente si verificano picchi nel traffico delle query. Specificando i livelli di promozione per le istanze serverless di DocumentDB in un cluster, è possibile configurare il cluster in modo che le istanze di lettura possano scalare indipendentemente dall'istanza di scrittura per gestire il carico aggiuntivo. Quando il picco di utilizzo diminuisce, le istanze del lettore vengono ridimensionate in modo da corrispondere alla capacità dell'istanza writer.

- Pianificazione della capacità: supponiamo che in genere si aggiusti la capacità del database o si verifichi la capacità ottimale del database per il carico di lavoro modificando le classi di istanze di tutte le istanze di un cluster. Con DocumentDB serverless, puoi evitare questo sovraccarico amministrativo. È possibile determinare la capacità minima e massima appropriata eseguendo il carico di lavoro e verificando la scalabilità effettiva delle istanze.
  - È possibile modificare le istanze esistenti da provisioned a DocumentDB serverless o da DocumentDB serverless a provisioned. In questi casi non è necessario creare un nuovo cluster o una nuova istanza.
- Sviluppo e test: oltre a eseguire le applicazioni più impegnative, puoi anche utilizzare DocumentDB serverless per ambienti di sviluppo e test. Con DocumentDB serverless, puoi creare istanze con una capacità minima bassa invece di utilizzare classi di istanze db.t* burstable. È possibile impostare una capacità massima sufficientemente elevata da consentire a tali istanze di eseguire carichi di lavoro considerevoli senza esaurire la memoria. Quando il database non è in uso, tutte le istanze vengono ridimensionate per evitare costi inutili.

Utilizzo di Amazon DocumentDB serverless per i carichi di lavoro con provisioning esistenti

Supponiamo di avere già un'applicazione DocumentDB in esecuzione su un cluster con provisioning. È possibile verificare come funzionerebbe l'applicazione con DocumentDB serverless aggiungendo una o più istanze serverless di DocumentDB al cluster esistente come istanze di lettura. È possibile verificare la frequenza con cui le istanze del lettore si ridimensionano verso l'alto e verso il basso. È possibile utilizzare il meccanismo di failover di DocumentDB per promuovere un'istanza serverless di DocumentDB come autore e verificare come gestisce il carico di lavoro. read/write Questo passaggio viene eseguito con tempi di inattività minimi e senza modificare l'endpoint utilizzato dalle applicazioni client. Per i dettagli sulla procedura per convertire i cluster esistenti in DocumentDB serverless, vedere. Migrazione ad Amazon DocumentDB serverless

Vantaggi di Amazon DocumentDB serverless

DocumentDB serverless è destinato a carichi di lavoro variabili o «impegnativi». A causa dei carichi di lavoro non prevedibili, potresti avere difficoltà a pianificare le modifiche della capacità del database. Potresti anche avere problemi ad apportare modifiche alla capacità con sufficiente rapidità utilizzando meccanismi familiari come l'aggiunta di istanze o la modifica delle classi di istanze. DocumentDB serverless offre i seguenti vantaggi per aiutare in questi casi d'uso:

- Gestione della capacità più semplice rispetto a quella fornita: DocumentDB serverless riduce lo sforzo di pianificazione delle dimensioni delle istanze e di ridimensionamento delle istanze al variare del carico di lavoro. Riduce inoltre lo sforzo necessario per mantenere una capacità costante per tutte le istanze di un cluster.

- Scalabilità più rapida e semplice durante i periodi di alta attività: DocumentDB scala senza server la capacità di calcolo e di memoria secondo necessità, senza interruzioni delle transazioni dei clienti o del carico di lavoro complessivo. La possibilità di utilizzare istanze di lettura con DocumentDB serverless consente di sfruttare il ridimensionamento orizzontale oltre a quello verticale. La possibilità di utilizzare i cluster globali di DocumentDB significa che è possibile distribuire il carico di lavoro di lettura serverless di DocumentDB su più server. Regioni AWS Questa funzionalità offre una praticità maggiore rispetto ai meccanismi di dimensionamento dei cluster con provisioning.

- Cost-effective durante i periodi di scarsa attività: DocumentDB serverless ti aiuta a evitare il provisioning eccessivo delle tue istanze. DocumentDB serverless aggiunge risorse in incrementi granulari quando le istanze si espandono. Paghi solo per le risorse di database che consumi. L'utilizzo delle risorse serverless di DocumentDB viene misurato al secondo. In questo modo, quando un'istanza viene ridimensionata, l'utilizzo ridotto delle risorse viene registrato immediatamente.

- Parità di funzionalità con provisioning: è possibile utilizzare tutte le funzionalità di DocumentDB con DocumentDB serverless. Ad esempio, con DocumentDB serverless è possibile utilizzare istanze di lettura, autenticazione del database AWS Identity and Access Management (IAM) e Performance Insights.

  In particolare, con DocumentDB serverless è possibile sfruttare le seguenti funzionalità dei cluster predisposti:

  - Istanze Reader: DocumentDB serverless può sfruttare le istanze di lettura per scalare orizzontalmente. Quando un cluster contiene una o più istanze di lettura, il cluster può eseguire il failover immediatamente in caso di problemi con l'istanza writer.
  - Multi-AZ cluster: è possibile distribuire le istanze serverless DocumentDB di un cluster su più zone di disponibilità (AZ). La configurazione di un Multi-AZ cluster aiuta a garantire la continuità aziendale anche nei rari casi di problemi che riguardano un intero AZ.

- Cluster globali: è possibile utilizzare DocumentDB serverless in combinazione con i cluster globali DocumentDB per creare copie aggiuntive di sola lettura del cluster in altri casi per scopi di disaster recovery. Regioni AWS

Come funziona Amazon DocumentDB serverless

Argomenti

- Panoramica di
- Configurazioni per i cluster Amazon DocumentDB
- Capacità di scalabilità serverless di Amazon DocumentDB
- Scalabilità senza server di Amazon DocumentDB
- Stato inattivo (0,5 DCU)

Panoramica di

Amazon DocumentDB serverless è adatto per i carichi di lavoro più impegnativi e altamente variabili. Ad esempio, l'uso del database potrebbe essere pesante per un breve periodo di tempo, seguito da lunghi periodi di attività leggera o nessuna attività. Alcuni esempi sono siti web per la vendita al dettaglio, i giochi o sportivi con eventi promozionali periodici e database in grado di generare report quando necessario. Altri sono ambienti di sviluppo e test e nuove applicazioni in cui l'utilizzo potrebbe aumentare rapidamente. In casi come questi e molti altri, non sempre è possibile configurare correttamente la capacità in anticipo con il modello fornito. Ciò può anche comportare costi più elevati se si esegue il provisioning eccessivo e si dispone di capacità che poi non viene utilizzata.

Al contrario, i cluster predisposti da DocumentDB sono adatti per carichi di lavoro costanti. Con i cluster con provisioning, si sceglie una classe di istanza con una quantità predefinita di memoria, potenza della CPU, larghezza di banda e così via. I/O Se il carico di lavoro cambia, modifichi manualmente la classe di istanza dello scrittore e dei lettori. Il modello soggetto a provisioning funziona bene quando è possibile regolare la capacità in anticipo rispetto ai modelli di consumo previsti ed è accettabile soffrire di brevi interruzioni mentre si modifica la classe di istanza dello scrittore e dei lettori all'interno del cluster.

DocumentDB serverless è progettato da zero per supportare cluster serverless che sono immediatamente scalabili. DocumentDB serverless è progettato per fornire lo stesso grado di sicurezza e isolamento degli scrittori e dei lettori predisposti. Questi aspetti sono cruciali negli

ambienti cloud serverless multitenant. Il meccanismo di dimensionamento dinamico impone un overhead molto ridotto in modo da poter rispondere rapidamente alle modifiche del carico di lavoro del database. È anche abbastanza potente da soddisfare i considerevoli aumenti della domanda di elaborazione.

Utilizzando DocumentDB serverless, è possibile creare un cluster DocumentDB senza essere vincolati a una capacità di database specifica per ogni scrittore e lettore. Puoi specificare l'intervallo minimo e massimo per la capacità. DocumentDB ridimensiona ogni scrittore o lettore serverless DocumentDB nel cluster entro tale intervallo di capacità. Utilizzando un Multi-AZ cluster in cui ogni scrittore o lettore può scalare dinamicamente, è possibile sfruttare il ridimensionamento dinamico e l'elevata disponibilità.

DocumentDB serverless ridimensiona automaticamente le risorse del database in base alle specifiche di capacità minima e massima. La scalabilità è rapida perché la maggior parte delle operazioni legate agli eventi di dimensionamento mantiene lo scrittore o il lettore sullo stesso host. Nei rari casi in cui uno scrittore o lettore serverless DocumentDB venga spostato da un host all'altro, DocumentDB serverless gestisce automaticamente le connessioni. Non è necessario modificare il codice dell'applicazione client del database o le stringhe di connessione al database.

Con DocumentDB serverless, come con i cluster con provisioning, la capacità di storage e la capacità di calcolo sono separate. Quando ci riferiamo alla capacità e alla scalabilità serverless di DocumentDB, è sempre la capacità di calcolo ad aumentare o diminuire. Pertanto, il cluster può contenere molti terabyte di dati anche quando la CPU e la capacità di memoria si dimensionano verso il basso.

Anziché effettuare il provisioning e gestire i server di database, puoi indicare la capacità del database. La capacità effettiva di ogni scrittore o lettore serverless DocumentDB varia nel tempo, a seconda del carico di lavoro. Per i dettagli su questi meccanismi, consulta Scalabilità senza server di Amazon DocumentDB.

Configurazioni per i cluster Amazon DocumentDB

Per ciascuno dei tuoi cluster Amazon DocumentDB, puoi scegliere qualsiasi combinazione di capacità serverless DocumentDB, capacità predisposta o entrambe.

Puoi configurare un cluster che contenga sia capacità serverless che provisioned di DocumentDB, chiamato cluster a configurazione mista. Ad esempio, supponiamo di aver bisogno di una read/write capacità maggiore di quella disponibile per un writer serverless DocumentDB. In questo caso puoi configurare il cluster con uno scrittore con provisioning di dimensioni molto ampie. Quindi puoi ancora

utilizzare DocumentDB serverless per i lettori. Oppure supponi che il carico di lavoro in scrittura per il cluster vari ma che il carico di lavoro in lettura sia costante. In questo caso, puoi configurare il tuo cluster con un writer serverless DocumentDB e uno o più lettori predisposti.

È inoltre possibile configurare un cluster in cui tutta la capacità è gestita da DocumentDB serverless. A tale scopo, è possibile creare un nuovo cluster e utilizzare DocumentDB serverless sin dall'inizio. Oppure puoi sostituire tutta la capacità fornita in un cluster esistente con DocumentDB serverless. Per le procedure per creare un nuovo cluster con DocumentDB serverless o per passare da un cluster esistente a DocumentDB serverless, vedere e. Creazione di un cluster che utilizza Amazon DocumentDB serverless Migrazione ad Amazon DocumentDB serverless

Se non si utilizza affatto DocumentDB serverless in un cluster, viene eseguito il provisioning di tutti gli autori e lettori del cluster. Questo è il tipo di cluster più comune con cui la maggior parte degli utenti ha familiarità. La capacità fornita è costante. Le tariffe sono relativamente semplici da prevedere. Tuttavia, è necessario prevedere in anticipo quanta capacità è necessaria. In alcuni casi le previsioni potrebbero essere imprecise o le esigenze di capacità potrebbero cambiare. In questi casi, il cluster può subire un sottoprovisioning (più lento del previsto) o un sovraprovisioning (più costoso di quanto si desideri).

Capacità di scalabilità serverless di Amazon DocumentDB

L'unità di misura per Amazon DocumentDB serverless è la DocumentDB Capacity Unit (DCU). La capacità di scalabilità serverless di DocumentDB non è legata alle classi di istanze utilizzate per i cluster con provisioning.

Ogni DCU è una combinazione di circa 2 gibibyte (GiB) di memoria, CPU corrispondente e rete. Puoi specificare l'intervallo di capacità del database utilizzando questa unità di misura. I parametri `ServerlessDatabaseCapacity` e `DCUUtilization` CloudWatch consentono di determinare la capacità effettivamente utilizzata dal database e se tale capacità rientra nell'intervallo specificato.

In qualsiasi momento, ogni scrittore o lettore serverless DocumentDB ha una capacità. La capacità è un numero a virgola mobile che rappresenta le DCU. La capacità aumenta o diminuisce ogni volta che lo scrittore o il lettore si dimensionano. Questo valore viene misurato ogni secondo. Per ogni cluster in cui si intende utilizzare DocumentDB serverless, si definisce un intervallo di capacità: i valori di capacità minima e massima tra cui ogni scrittore o lettore serverless DocumentDB può scalare. L'intervallo di capacità è lo stesso per ogni scrittore o lettore serverless DocumentDB in un cluster. Ogni scrittore o lettore serverless DocumentDB ha una propria capacità, che rientra da qualche parte in tale intervallo.

DocumentDB serverless è supportato su DocumentDB 5.0 e 8.0 con un intervallo di capacità compreso tra 0,5 e 256 DCU.

La capacità serverless di DocumentDB più piccola che è possibile definire è 0,5 DCU. È possibile specificare un numero più alto se è inferiore o uguale al valore di capacità massima supportata. L'impostazione della capacità minima su un numero ridotto consente ai cluster leggermente caricati di consumare risorse di calcolo minime. Allo stesso tempo, rimangono pronti ad accettare immediatamente le connessioni e a dimensionarsi quando diventano impegnati.

Impostate il valore minimo su un valore che consenta a ciascun writer o lettore di contenere il working set dell'applicazione nel buffer pool. In questo modo, il contenuto del buffer pool non viene scartato durante i periodi di inattività. Per tutte le considerazioni sulla scelta dell'intervallo di capacità di scalabilità, vedere. Scelta dell'intervallo di capacità di scalabilità per un cluster serverless DocumentDB

A seconda di come configurate i lettori in una Multi-AZ distribuzione, le loro capacità possono essere legate alla capacità dello scrittore o indipendentemente. Per i dettagli su come eseguire queste operazioni, consulta Visualizzazione e modifica del livello di promozione dei lettori serverless.

Il monitoraggio serverless di DocumentDB implica la misurazione dei valori di capacità del writer e dei lettori del cluster nel tempo. Se il database non viene si dimensiona verso il basso fino alla capacità minima, puoi intraprendere azioni come la regolazione del minimo e l'ottimizzazione dell'applicazione database. Se il database raggiunge costantemente la sua capacità massima, puoi intraprendere operazioni come l'aumento di tale vincolo. Puoi inoltre ottimizzare l'applicazione di database e distribuire il carico di query su più lettori.

I costi per la capacità serverless di DocumentDB sono misurati in termini di. DCU-hours Per informazioni su come vengono calcolati i costi serverless di DocumentDB, consulta i prezzi di Amazon DocumentDB. Supponiamo che il numero totale di istanze di scrittura e di lettura nel cluster sia n. In tal caso, il cluster consuma circa n x DCU minime quando non si eseguono operazioni di database. Amazon DocumentDB stesso potrebbe eseguire operazioni di monitoraggio o manutenzione che causano una piccola quantità di carico. Questo cluster non consuma più di n x unità DCU massime quando il database è in esecuzione a piena capacità.

Per ulteriori dettagli sulla scelta dei valori DCU minimi e massimi appropriati, vedere. Scelta dell'intervallo di capacità di scalabilità per un cluster serverless DocumentDB I valori DCU minimi e massimi specificati influiscono anche su alcuni limiti delle istanze Amazon DocumentDB. Per dettagli sull'interazione tra l'intervallo di capacità e i limiti delle istanze, consulta. Limiti delle istanze serverless di Amazon DocumentDB

Scalabilità senza server di Amazon DocumentDB

Per ogni scrittore o lettore serverless DocumentDB, Amazon DocumentDB monitora continuamente l'utilizzo di risorse come CPU, memoria e rete. Queste misurazioni sono chiamate collettivamente carico. Il carico include le operazioni di database eseguite dall'applicazione. Include anche l'elaborazione in background per il server del database e le attività amministrative di Amazon DocumentDB. Quando la capacità è limitata da una di queste condizioni, DocumentDB è scalabile senza server. DocumentDB serverless aumenta anche la scalabilità quando rileva problemi di prestazioni che può risolvere in tal modo. È possibile monitorare l'utilizzo delle risorse e il modo in cui influisce sulla scalabilità serverless di DocumentDB utilizzando le procedure in. Monitoraggio di Amazon DocumentDB senza server

Il carico può variare a seconda dello scrittore e dei lettori del cluster. Il writer gestisce le operazioni di scrittura ed esegue tutte le modifiche ai dati sul volume del cluster. I lettori possono elaborare richieste di sola lettura.

La scalabilità è l'operazione che aumenta o diminuisce la capacità serverless di DocumentDB per il database. Con DocumentDB serverless, ogni scrittore e lettore ha il proprio valore di capacità attuale, misurato in DCU. DocumentDB serverless scala uno scrittore o un lettore fino a una capacità superiore quando la sua capacità attuale è troppo bassa per gestire il carico. Ridimensiona lo scrittore o il lettore a una capacità inferiore quando la sua capacità corrente è superiore a quella necessaria.

DocumentDB serverless può aumentare la capacità in modo incrementale. Quando la richiesta di carico di lavoro inizia a raggiungere l'attuale capacità del database di uno scrittore o lettore, DocumentDB serverless aumenta il numero di DCU per tale scrittore o lettore. DocumentDB serverless scala la capacità negli incrementi necessari per fornire le migliori prestazioni per le risorse consumate. La scalabilità avviene con incrementi fino a 0,5 DCU. Maggiore è la capacità attuale, maggiore è l'incremento nel dimensionamento e quindi più velocemente può essere garantito il dimensionamento.

Poiché il ridimensionamento serverless di DocumentDB è così frequente, granulare e senza interruzioni, non causa eventi discreti nel. Console di gestione AWS Invece, puoi misurare le CloudWatch metriche di Amazon come `serverlessDatabaseCapacity` e e monitorare i loro valori minimi`DCUUtilization`, massimi e medi nel tempo. Per saperne di più sul monitoraggio di DocumentDB serverless, consulta. Monitoraggio di Amazon DocumentDB senza server

Il ridimensionamento verso l'alto o verso il basso può essere causato da quanto segue:

- Utilizzo della memoria
- Utilizzo CPU
- Utilizzo rete
- Utilizzo dell'archiviazione

È possibile monitorare queste cause di ridimensionamento up/down sulle istanze serverless di DocumentDB. Per ulteriori informazioni, consulta Monitoraggio di Amazon DocumentDB senza server.

Puoi scegliere di creare un dimensionamento del lettore in modo contemporaneo allo scrittore associato o indipendentemente da questo. Tale obiettivo si raggiunge specificando il livello di promozione per quel lettore.

- I lettori serverless di DocumentDB, nei livelli di promozione 0 e 1, sono scalabili contemporaneamente allo scrittore. Questo comportamento di dimensionamento rende i lettori con livelli prioritari 0 e 1 ideali per la disponibilità. Questo perché sono sempre dimensionati alla capacità giusta per assumere il carico di lavoro dallo scrittore in caso di failover.
- I lettori nei livelli di promozione da 2 a 15 si dimensionano indipendentemente dallo scrittore. Ogni lettore rimane entro i valori DCU minimi e massimi specificati per il cluster. Quando un lettore si dimensiona indipendentemente dallo scrittore del database associato, può diventare inattivo e dimensionarsi verso il basso mentre lo scrittore continua a elaborare un volume elevato di transazioni. Se nessun altro lettore è disponibile in livelli di promozione inferiori rimane ancora disponibile come target di failover. Tuttavia, se viene promosso al ruolo di scrittore, potrebbe essere necessario un dimensionamento verso l'alto per gestire l'intero carico di lavoro dello scrittore.

Per informazioni dettagliate sulla visualizzazione e la modifica dei livelli di promozione delle istanze serverless, consulta. Visualizzazione e modifica del livello di promozione dei lettori serverless

La scalabilità serverless di DocumentDB può avvenire mentre le connessioni al database sono aperte, mentre le transazioni sono in corso, ecc. DocumentDB serverless non aspetta un momento di silenzio per iniziare la scalabilità. Il dimensionamento non interrompe le operazioni del database in corso.

Se il carico di lavoro richiede una capacità di lettura superiore a quella disponibile con un singolo writer e un singolo lettore, puoi aggiungere più lettori serverless DocumentDB al cluster. Ogni lettore serverless DocumentDB può scalare entro l'intervallo di valori di capacità minima e massima

specificati per il cluster. Puoi utilizzare l'endpoint di lettura del cluster per gestire le sessioni di sola lettura attraverso i lettori e ridurre il carico sullo scrittore.

Il fatto che DocumentDB esegua il ridimensionamento senza server e la velocità con cui si verifica il ridimensionamento una volta avviato dipende anche dalle impostazioni DCU minime e massime per il cluster. Inoltre, dipendono dal fatto che un lettore sia configurato per dimensionarsi contemporaneamente allo scrittore o indipendentemente da esso. Per dettagli sulla configurazione del ridimensionamento, vedere. Configurazione di scalabilità serverless di Amazon DocumentDB

Stato inattivo (0,5 DCU)

Quando gli scrittori o lettori serverless di Amazon DocumentDB sono inattivi, le istanze serverless di DocumentDB supportano la scalabilità fino a uno stato di inattività di 0,5 DCU se il cluster è configurato per essere 0,5. MinCapacity

Nello stato inattivo, le istanze serverless DocumentDB non dispongono di una capacità di elaborazione della CPU sufficiente per supportare la maggior parte dei carichi di lavoro di produzione, ma sono pronte per essere scalate rapidamente per supportare un nuovo carico di lavoro. In uno stato non inattivo, le istanze serverless di DocumentDB richiedono in genere almeno 1,0 - 2,5 DCU. Pertanto, quando le istanze serverless di DocumentDB passano da uno stato inattivo a uno stato non inattivo, saliranno direttamente a 1,0 - 2,5 DCU (o al valore di se è inferiore). MaxCapacity

Per supportare la scalabilità fino a 0,5 DCU quando sono inattive, i limiti delle istanze sono limitati se sono configurate per essere inferiori o uguali a 1,0 DCU. MinCapacity Per ulteriori informazioni su come i limiti sono influenzati dalla configurazione, vedere. MinCapacity Limiti delle istanze serverless di Amazon DocumentDB

Requisiti e limitazioni per DocumentDB serverless

Requisiti delle funzionalità

Disponibilità nelle regioni

Per ulteriori informazioni su Regioni AWS dove è disponibile il tipo di istanza serverless di Amazon DocumentDB, consulta la colonna Serverless in. Classi di istanze supportate per regione

Il AWS CLI comando seguente può essere utilizzato per verificare le esatte opzioni di istanza serverless di DocumentDB offerte in una particolare regione:

```
aws docdb describe-orderable-db-instance-options \
        --region my_region \
        --db-instance-class db.serverless \
        --engine docdb
```

Disponibilità della versione del motore

DocumentDB serverless è supportato dalle versioni del motore Amazon DocumentDB 5.0 e 8.0. Non è disponibile nelle versioni del motore 3.6 o 4.0.

> **Note**
>
> DocumentDB serverless è supportato solo nelle versioni di patch più recenti della 5.0.0. Assicurati che il cluster sia aggiornato a una versione recente della patch del motore. Per ulteriori informazioni sulla manutenzione dei cluster Amazon DocumentDB, consulta Manutenzione di Amazon DocumentDB

Configurazione del cluster

Prima di aggiungere qualsiasi istanza serverless di Amazon DocumentDB a un cluster Amazon DocumentDB, anche il cluster deve avere il set di parametri. `ServerlessV2ScalingConfiguration` Questo definisce l'intervallo di capacità delle istanze serverless DocumentDB nel cluster. Per ulteriori informazioni sulla configurazione della scalabilità, vedere. Configurazione di scalabilità serverless di Amazon DocumentDB

Impostazioni minime dell'intervallo di capacità di scalabilità per alcune funzionalità di Amazon DocumentDB

Alcune funzionalità di Amazon DocumentDB funzionano con DocumentDB senza server, ma potrebbero causare problemi se l'intervallo di capacità è inferiore a quello necessario per i requisiti di memoria per tali funzionalità con un carico di lavoro specifico. In tal caso, il database potrebbe non funzionare come al solito o potrebbero verificarsi errori dovuti all'esaurimento della memoria.

Le seguenti funzionalità richiedono la configurazione di un `MinCapacity` and/or `MaxCapacity` valore più elevato per un funzionamento ottimale:

- Approfondimenti sulle prestazioni

- Cluster globali (si applica solo al primario Regione AWS)
- Creazione di istanze serverless su un cluster con un grande volume di dati

  Ciò include la creazione di istanze serverless come parte di un ripristino del cluster.

Per consigli sull'impostazione dell'intervallo di capacità appropriato (se si utilizza questa funzionalità), vedereScelta dell'intervallo di capacità di scalabilità per un cluster serverless DocumentDB. Per informazioni sulla risoluzione dei problemi quando il database rileva errori di memoria insufficiente causati da un intervallo di capacità configurato in modo errato, consulta Esclusione di errori dovuti all'esaurimento della memoria.

Creazione di un cluster che utilizza Amazon DocumentDB serverless

Creazione di un cluster serverless Amazon DocumentDB

Con Amazon DocumentDB serverless, i tuoi cluster sono intercambiabili con i cluster con provisioning. Puoi avere cluster in cui alcune istanze utilizzano il serverless e altre il provisioning.

Verifica che la regione e la versione del motore desiderate supportino DocumentDB serverless. Consulta Requisiti e limitazioni per DocumentDB serverless.

Per creare un cluster Amazon DocumentDB in cui aggiungere istanze serverless, segui la stessa procedura descritta in. Creazione di un cluster Amazon DocumentDB L'unica differenza è che deve essere fornito anche l'`ServerlessV2ScalingConfiguration`argomento.

L'`ServerlessV2ScalingConfiguration`argomento specifica l'intervallo di capacità di scalabilità delle istanze serverless di DocumentDB. È costituito dai valori minimi e massimi dell'unità di capacità DocumentDB (DCU) che si applicano a tutte le istanze serverless di DocumentDB nel cluster:

- Il `MinCapacity` valore specifica la capacità di scalabilità minima.
- Il `MaxCapacity` valore specifica la capacità di scalabilità massima.

Per ulteriori informazioni sul dimensionamento, consultare Configurazione di scalabilità serverless di Amazon DocumentDB.

Using the Console di gestione AWS

Il seguente esempio Console di gestione AWS di configurazione mostra come creare un cluster serverless DocumentDB.

1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.
2. Nel pannello di navigazione scegliere Cluster.

> **Tip**
>
> Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

   Viene visualizzata la tabella Clusters.
3. Scegli Crea.

   Viene visualizzata la pagina Crea cluster Amazon DocumentDB.
4. Nella pagina Crea cluster Amazon DocumentDB, nella sezione Tipo di cluster, scegli Instance-based cluster (questa è l'opzione predefinita).
5. Nella sezione Configurazione del cluster:
   a. Per l'identificatore del cluster, inserisci un nome univoco, ad esempio**`myserverlesscluster`**. Tieni presente che la console cambierà tutti i nomi dei cluster in lettere minuscole indipendentemente da come vengono immessi.
   b. Per la versione Engine, scegli 5.0.0 (questa è l'opzione predefinita).
6. Nella sezione di configurazione dello storage del cluster, scegli Amazon DocumentDB Standard (questa è l'opzione predefinita).

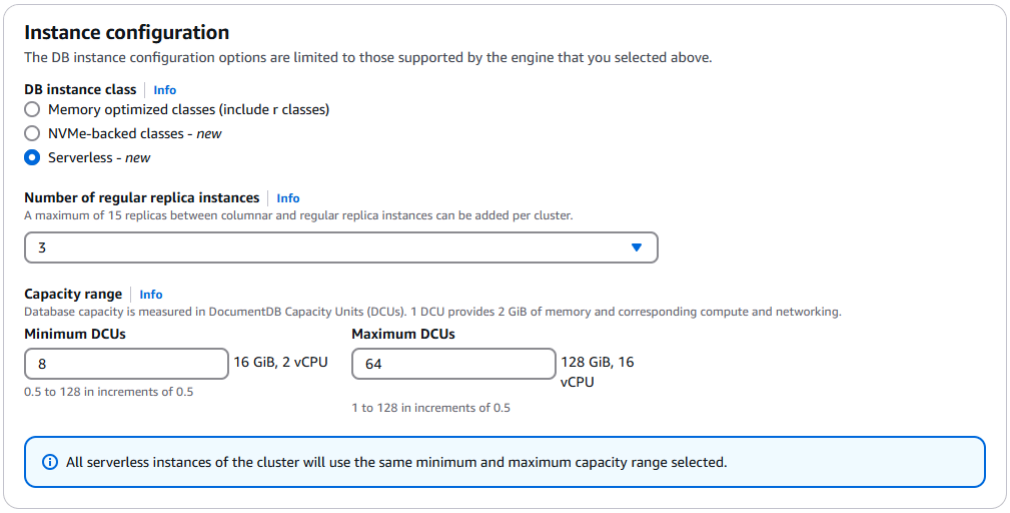
> **Note**
>
> L'altra opzione in questa categoria è Amazon I/O-Optimized DocumentDB. Per ulteriori informazioni su entrambe le opzioni, consulta Configurazioni di storage del cluster Amazon DocumentDB

7. Nella sezione Configurazione dell'istanza:

a. Per Classe di istanza database scegli Serverless.

b. Per Numero di istanze di replica regolari, scegli 3 (questa è l'opzione predefinita).

c. Nella sezione Intervallo di capacità, lascia i valori predefiniti per DCU minime e DCU massime. Per informazioni sull'impostazione di questi parametri, vedere. Limiti delle istanze serverless di Amazon DocumentDB



8. Nella sezione Connettività, lascia l'impostazione predefinita di Non connetterti a una risorsa di calcolo EC2.

9. Nella sezione Autenticazione, inserisci un nome utente per l'utente principale, quindi scegli Gestito autonomamente. Inserisci una password, quindi confermala.

   Se invece hai scelto Managed in AWS Secrets Manager, consulta Gestione delle password con Amazon DocumentDB e AWS Secrets Manager per ulteriori informazioni.

10. Lascia tutte le altre opzioni come predefinite e scegli Crea cluster.

Using the AWS CLI

Nell'esempio seguente, sostituisci ciascuna *`user input placeholder`* con le tue informazioni o i tuoi parametri di configurazione.

Per creare un cluster compatibile con le istanze serverless di DocumentDB utilizzando, vedere. AWS CLICreazione di un cluster utilizzando AWS CLI

Includi i seguenti parametri aggiuntivi nel tuo comando: `create-db-cluster`

```
--serverless-v2-scaling-configuration
     MinCapacity=minimum_capacity,MaxCapacity=maximum_capacity
```

Esempio:

```
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
```

Aggiungere un'istanza serverless di Amazon DocumentDB

Per aggiungere un'istanza serverless DocumentDB, segui la stessa procedura inAggiungere un'istanza Amazon DocumentDB a un cluster, assicurandoti di specificare db.serverless come classe di istanza.

Aggiungere un'istanza serverless utilizzando Console di gestione AWS.

Per aggiungere un'istanza serverless di DocumentDB utilizzando la console, consulta Aggiungere un'istanza Amazon DocumentDB a un cluster e scegli la scheda Usare la. Console di gestione AWS

Aggiungere un'istanza serverless utilizzando AWS CLI

Per aggiungere un'istanza serverless di DocumentDB utilizzando il AWS CLI, consulta Aggiungere un'istanza Amazon DocumentDB a un cluster e scegli la scheda Utilizzo del. AWS CLI

Utilizzate il seguente argomento CLI della classe di istanza:

```
--db-instance-class db.serverless
```

Esempio:

```
aws docdb create-db-instance \
      --db-cluster-identifier sample-cluster \
      --db-instance-identifier sample-instance \
```

```
    --db-instance-class db.serverless \
    --engine docdb
```

Migrazione ad Amazon DocumentDB serverless

Argomenti

- Migrazione dei cluster DocumentDB esistenti a quelli serverless
- Migrazione da MongoDB a DocumentDB serverless

Migrazione dei cluster DocumentDB esistenti a quelli serverless

Aggiornamento della versione del motore di un cluster

Se il cluster di cui hai eseguito il provisioning esegue una versione inferiore del motore che non supporta DocumentDB serverless, devi prima aggiornare il cluster a una versione del motore supportata. Per ulteriori informazioni, consulta Aggiornamento della versione principale in-place di Amazon DocumentDB.

Migrazione di un cluster con provisioning su DocumentDB serverless

Per passare da un cluster con provisioning a utilizzare DocumentDB serverless, segui questi passaggi:

1. Verifica se la versione del motore del cluster fornito deve essere aggiornata per essere utilizzata con DocumentDB serverless. Consulta Requisiti e limitazioni per DocumentDB serverless.

   > **Note**
   >
   > Se il cluster fornito esegue una versione del motore che non è disponibile per DocumentDB serverless, aggiorna la versione del motore del cluster. Consulta Aggiornamento della versione principale in-place di Amazon DocumentDB.

2. Definire la configurazione di dimensionamento per il cluster. Per i dettagli sulla scelta della configurazione di scalabilità, vedere. Scelta dell'intervallo di capacità di scalabilità per un cluster serverless DocumentDB Per modificare la configurazione di scalabilità per un cluster, vedere. Visualizzazione e modifica della configurazione dell'intervallo di capacità di scalabilità di un cluster

3. Configura qualsiasi altra proprietà del cluster per tenere conto dei requisiti e delle limitazioni serverless di DocumentDB da. Requisiti e limitazioni per DocumentDB serverless
4. Aggiungi una o più istanze serverless di DocumentDB al cluster. Segui la procedura riportata in Aggiungere un'istanza serverless di Amazon DocumentDB.

   > **Note**
   >
   > In alcuni casi, è possibile che nel cluster siano già presenti una o più istanze di lettura predisposte. In tal caso, puoi scegliere di convertire uno dei lettori in un'istanza serverless di DocumentDB invece di creare una nuova istanza. A tale scopo, segui la procedura in Modifica della classe di un'istanza.

5. (Facoltativo) Esegui un'operazione di failover per rendere un'istanza serverless di DocumentDB il cluster writer. Consulta Failover di Amazon DocumentDB.
6. (Facoltativo) Converti tutte le istanze Amazon DocumentDB con provisioning rimanenti in istanze serverless DocumentDB (vediModifica della classe di un'istanza) o rimuovile dal cluster (vedi). Eliminazione di un'istanza Amazon DocumentDB

Using the Console di gestione AWS

L'esempio di Console di gestione AWS configurazione seguente mostra il processo di migrazione utilizzando un cluster con provisioning di Amazon DocumentDB su cui è in esecuzione Amazon DocumentDB 5.0.0, che non richiede un aggiornamento della versione del motore per iniziare a utilizzare DocumentDB serverless. Il cluster è denominato e inizia con tre istanze predisposte denominate`sample`, e. `sample` `sample2` `sample3` In questo esempio, queste tre istanze verranno sostituite da tre istanze serverless.

1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.
2. Nel pannello di navigazione scegliere Cluster.

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

Viene visualizzata la tabella Clusters.

3. Nella tabella Clusters, seleziona la casella di controllo del cluster a cui desideri aggiungere un'istanza serverless.
4. Scegli Actions (Operazioni), quindi Add instance (Aggiungi istanza).
5. Nella finestra di dialogo Aggiungi istanze, seleziona Serverless nella sezione della classe di istanza DB per ogni nuova istanza serverless che desideri creare.
6. Per le impostazioni della capacità serverless, imposta la configurazione di scalabilità in base alla descrizione della capacità nella finestra di dialogo.
7. (Facoltativo) Per aggiungere un'altra istanza, scegli Aggiungi istanza. Continua ad aggiungere istanze fino a raggiungere il numero desiderato di nuove istanze.

   In questo esempio, vengono create tre nuove istanze serverless.
8. Scegli Crea.

   La creazione dell'istanza richiede diversi minuti. È possibile utilizzare la console o la AWS CLI per visualizzare lo stato dell'istanza. Per ulteriori informazioni, consulta Monitoraggio dello stato di un cluster Amazon DocumentDB.
9. Tornando alla tabella Clusters, seleziona le caselle di controllo di tutte e tre le istanze originariamente assegnate.
10. Scegli Azioni, quindi Elimina.

Durante l'eliminazione, viene eseguito automaticamente un failover per promuovere una delle istanze rimanenti come writer. Dopo alcuni minuti, il processo di eliminazione è completato. Il cluster esistente ora contiene tre istanze serverless di DocumentDB (come definito nella colonna Dimensioni).

Using the AWS CLI

L'esempio di AWS CLI configurazione seguente mostra il processo di migrazione utilizzando un cluster fornito da Amazon DocumentDB su cui è in esecuzione Amazon DocumentDB 5.0.0, che non richiede un aggiornamento della versione del motore per iniziare a utilizzare DocumentDB serverless. Il cluster è denominato e inizia con tre istanze predisposte denominate`sample`, e. `sample` `sample2` `sample3` In questo esempio, queste tre istanze verranno sostituite da tre istanze serverless. Il cluster è denominato `sample-cluster` e inizia con due istanze assegnate

denominate `sample-provisioned-instance-1` and`sample-provisioned-instance-2`, un'istanza writer e un'istanza reader.

Nell'esempio seguente, sostituisci ciascuna *`user input placeholder`* con le tue informazioni o i tuoi parametri di configurazione.

Utilizzate l'`aws docdb describe-db-clusters`operazione per determinare lo stato di un cluster. Il codice seguente trova lo stato del cluster `sample-cluster` e restituisce i risultati in una tabella:

```
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             |
+---------------------------------+---------------------------------
+------------------+----------------+
```

Aggiungi la configurazione di scalabilità per il cluster:

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16
```

Aggiungi le istanze serverless. In questo esempio, vengono aggiunte nuove istanze serverless `sample-serverless-instance-1` denominate e: `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
```

Inserisci quanto segue per attendere che le istanze serverless siano disponibili prima di procedere:

```
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
```

Eseguite un failover per rendere la nuova `sample-serverless-instance-1` istanza il cluster writer:

```
aws docdb failover-db-cluster \
    --db-cluster-identifier sample-cluster \
    --target-db-instance-identifier sample-serverless-instance-1
```

Il completamento del failover richiede alcuni secondi, dopodiché sample-serverless-instance-1 diventa il cluster writer. Verificalo con il seguente input:

```
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             |
+--------------------------------+---------------------------------
+-----------------+---------------+
```

Infine, elimina le istanze originali assegnate:

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-provisioned-instance-1

aws docdb delete-db-instance \
    --db-instance-identifier sample-provisioned-instance-2
```

Migrazione da MongoDB a DocumentDB serverless

Puoi migrare i tuoi database MongoDB su DocumentDB serverless, proprio come con Amazon DocumentDB con provisioning. Per ulteriori informazioni, consulta Migrazione e aggiornamento di Amazon DocumentDB.

Gestione di Amazon DocumentDB senza server

Visualizzazione e modifica della configurazione dell'intervallo di capacità di scalabilità di un cluster

L'`ServerlessV2ScalingConfiguration`argomento specifica l'intervallo di capacità di scalabilità delle istanze serverless di DocumentDB. È costituito dai valori minimi e massimi dell'unità di capacità DocumentDB (DCU) che si applicano a tutte le istanze serverless di DocumentDB nel cluster.

- **`MinCapacity`**— La capacità di scalabilità minima di qualsiasi istanza serverless di DocumentDB nel cluster.
- **`MaxCapacity`**— La capacità di scalabilità massima di qualsiasi istanza serverless di DocumentDB nel cluster.

> **Note**
>
> Le seguenti modifiche alla configurazione di scalabilità richiedono il riavvio dell'istanza per riflettere i nuovi valori: `MinCapacity` `MaxCapacity`
>
> - Eventuali modifiche al valore `MaxCapacity`
> - Modifica del `MinCapacity` valore da un valore superiore a 1,0
> - Modifica del `MinCapacity` valore a maggiore di 1,0 da un valore inferiore

Per ulteriori informazioni sulla configurazione di scalabilità e su come selezionare i limiti di capacità di scalabilità appropriati, vedere. Configurazione di scalabilità serverless di Amazon DocumentDB

Using the Console di gestione AWS

Il seguente esempio Console di gestione AWS di configurazione mostra come visualizzare e modificare le impostazioni di configurazione di scalabilità di un cluster serverless DocumentDB.

1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.
2. Nel pannello di navigazione scegliere Cluster.

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

   Viene visualizzata la tabella Clusters.
3. Nella tabella Clusters, seleziona la casella di controllo del cluster di cui desideri modificare la capacità di scalabilità.

4. Scegli Actions (Operazioni), quindi Modify (Modifica).

   Viene visualizzata la finestra di dialogo Modifica cluster.

5. Individua la sezione di impostazione della capacità Serverless e imposta la configurazione di ridimensionamento (intervallo di capacità) in base alla descrizione della capacità nella finestra di dialogo.

   Per ulteriori informazioni sulla scalabilità e sugli intervalli di capacità, vedere. Configurazione di scalabilità serverless di Amazon DocumentDB

6. Scegli Continue (Continua).
7. In Scheduling of Modifications (Pianificazione delle modifiche), scegliere Apply immediately (Applica immediatamente).
8. Scegliere Modify cluster (Modifica cluster).
9. Una volta completata la modifica, è necessario riavviare ogni istanza serverless. Per ridurre al minimo l'indisponibilità dello scrittore, eseguite la seguente sequenza di operazioni:

   a. Riavviate ogni istanza del lettore serverless.

      i. Seleziona l'istanza del lettore, scegli Azioni, quindi Riavvia.

      ii. Attendi che lo stato dell'istanza ritorni su Disponibile.

   b. Esegui un failover su un'istanza serverless riavviata.

      i. Seleziona il cluster, scegli Azioni, quindi Failover.

      ii. Attendi il completamento dell'operazione di failover.

   c. Riavvia l'istanza serverless rimanente.

      i. Seleziona l'istanza rimanente, scegli Azioni, quindi Riavvia.

      ii. Attendi che lo stato dell'istanza ritorni su Disponibile.

Using the AWS CLI

L'esempio AWS CLI di configurazione seguente mostra la configurazione di ridimensionamento corrente.

Nel seguente esempio, sostituite ciascuno di essi *user input placeholder* con informazioni o parametri di configurazione personalizzati.

L'attuale configurazione di scalabilità del cluster può essere visualizzata utilizzando il `describe-db-clusters` AWS CLI comando:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[0].ServerlessV2ScalingConfiguration'
```

L'output di questo comando è il seguente:

```
{
    "MinCapacity": 0.5,
    "MaxCapacity": 16.0
}
```

La configurazione di scalabilità del cluster può essere modificata utilizzando il `modify-db-cluster` comando:

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=32
```

Una volta completata, ogni istanza serverless deve essere riavviata. Per ridurre al minimo l'indisponibilità dello scrittore, possiamo eseguire la seguente sequenza di operazioni:

```
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
```

Visualizzazione e modifica del livello di promozione dei lettori serverless

Per i cluster contenenti più istanze serverless di DocumentDB o una combinazione di istanze predisposte e serverless, presta attenzione all'impostazione del livello di promozione per ciascuna istanza serverless. Questa impostazione controlla un comportamento maggiore per le istanze serverless rispetto alle istanze con provisioning.

Per le istanze con provisioning, la scelta del livello 0-15 determina solo l'ordine in cui Amazon DocumentDB sceglie l'istanza del lettore da promuovere al writer durante un'operazione di failover. Tuttavia, per le istanze serverless, il numero di livello determina anche se l'istanza si espande in modo da corrispondere alla capacità dell'istanza di scrittura o se si ridimensiona in modo indipendente in base al proprio carico di lavoro. Le istanze di lettura serverless di livello 0 o 1 sono mantenute a una capacità minima almeno pari a quella dell'istanza writer. In questo modo, sono pronte a sostituire l'istanza writer in caso di failover. Se l'istanza writer è un'istanza con provisioning, Amazon DocumentDB stima la capacità serverless equivalente di DocumentDB. Utilizza tale stima come capacità minima per l'istanza del lettore serverless.

Le istanze di lettura serverless DocumentDB nei livelli 2—15 non hanno lo stesso vincolo sulla loro capacità minima. Quando sono inattive, possono essere ridimensionate fino al valore minimo dell'unità di capacità (DCU) di DocumentDB specificato nell'intervallo di capacità del cluster.

Using the Console di gestione AWS

Il seguente esempio Console di gestione AWS di configurazione mostra come visualizzare e modificare le impostazioni del livello promozionale di un lettore di istanze serverless DocumentDB.

1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.

   I livelli di promozione di ciascuna istanza sono visualizzati nella colonna Livello di promozione in. Console di gestione AWS
2. Nel pannello di navigazione scegliere Cluster.

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu

(≡) nell'angolo in alto a sinistra della pagina.

Viene visualizzata la tabella Clusters.

3. Nella tabella Clusters, seleziona la casella di controllo dell'istanza in cui desideri modificare il livello di promozione.
4. Scegli Actions (Operazioni), quindi Modify (Modifica).

   Viene visualizzata la finestra di dialogo Modifica istanza.
5. Individua la sezione Failover e imposta il livello di promozione sul livello desiderato.
6. Scegli Continue (Continua).
7. In Scheduling of Modifications (Pianificazione delle modifiche), scegliere Apply immediately (Applica immediatamente).
8. Scegli Modify instance (Modifica istanza).

Using the AWS CLI

L'esempio di AWS CLI configurazione seguente mostra la configurazione di scalabilità corrente.

Nel seguente esempio, sostituite ciascuno di essi *`user input placeholder`* con informazioni o parametri di configurazione personalizzati.

I livelli di promozione di tutte le istanze di un cluster possono essere visualizzati utilizzando il `describe-db-clusters` AWS CLI comando:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[0].DBClusterMembers' \
    --output table
```

L'output di questo comando è il seguente:

```
----------------------------------------------------------------------------------------------
|                                        DescribeDBClusters                                  |
                  |
+---------------------------------+---------------------------------
+------------------+----------------+
```

```
|  DBClusterParameterGroupStatus |      DBInstanceIdentifier       | IsClusterWriter
  | PromotionTier  |
+--------------------------------+---------------------------------
+------------------+---------------+
|  in-sync                       |  sample-serverless-instance-2   |  False
  |  1             |
|  in-sync                       |  sample-serverless-instance-1   |  True
  |  1             |
+--------------------------------+---------------------------------
+------------------+---------------+
```

Il livello di promozione di un'istanza specifica può essere modificato utilizzando il `modify-db-instance` comando:

```
aws docdb modify-db-instance \
    --db-instance-identifier sample-serverless-instance-2 \
    --promotion-tier 3
```

Limiti delle istanze serverless di Amazon DocumentDB

Per le istanze serverless DocumentDB, i seguenti limiti per istanza dipendono dalla capacità di scalabilità attuale dell'istanza:

- Memoria di istanza (GiB)
- Connessioni (tutte)
- Limite del cursore
- Transazioni aperte
- Connessioni (attive)

Le tabelle seguenti descrivono in che modo i limiti per istanza delle istanze serverless si ridimensionano in relazione alla capacità di scalabilità corrente dell'istanza.

Nota che i valori limite sono diversi a seconda che la configurazione di `MinCapacity` scalabilità del cluster sia stata impostata su un valore maggiore di due (2). Se `MinCapacity` è impostato su un valore inferiore o uguale a 2, i valori limite per il limite del cursore, le transazioni aperte e le connessioni (attive) sono limitati a un valore massimo inferiore. (Il motivo tecnico del limite è che supporta il ridimensionamento a uno stato «inattivo». Per ulteriori informazioni, consulta Stato inattivo (0,5 DCU).

limiti di istanza db.serverless se ≤1 MinCapacity

Capacità attuale (DCU)	Memoria di istanza (GiB)	Connessioni (tutte)	Limite del cursore	Transazioni aperte	Connessioni (attive)
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
256	512	60000	132	264	1550

limiti di istanza db.serverless se >1 MinCapacity

Capacità attuale (DCU)	Memoria di istanza (GiB)	Connessioni (tutte)	Limite del cursore	Transazioni aperte	Connessioni (attive)
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

Capacità attuale (DCU)	Memoria di istanza (GiB)	Connessioni (tutte)	Limite del cursore	Transazioni aperte	Connessioni (attive)
32	64	16000	384	768	4500
64	128	32000	768	1536	4500
128	256	60000	1536	3072	4500
256	512	60000	3072	6144	4500

Puoi monitorare e segnalare i limiti per istanza utilizzando le seguenti CloudWatch metriche. Per ulteriori informazioni sulle CloudWatch metriche di Amazon DocumentDB, consulta. Monitoraggio di Amazon DocumentDB con CloudWatch

Risorsa	CloudWatch metrica limite	CloudWatch metrica di utilizzo (massimo 1 minuto)	CloudWatch metrica di utilizzo
Memoria di istanza	-	-	FreeableMemory
Connessioni (tutte)	DatabaseConnectionsLimit	DatabaseConnectionsMax	DatabaseConnections
Cursori	DatabaseCursorsLimit	DatabaseCursorsMax	DatabaseCursors
Transazioni	TransactionsOpenLimit	TransactionsOpenMax	TransactionsOpen

Configurazione di scalabilità serverless di Amazon DocumentDB

Argomenti

- Scelta dell'intervallo di capacità di scalabilità per un cluster serverless DocumentDB
- Scelta dell'MinCapacityimpostazione per un cluster serverless DocumentDB
- Scelta dell'MaxCapacityimpostazione per un cluster serverless DocumentDB
- Esclusione di errori dovuti all'esaurimento della memoria

- Perché la mia istanza serverless non viene ridimensionata?

Scelta dell'intervallo di capacità di scalabilità per un cluster serverless DocumentDB

Prima di aggiungere qualsiasi istanza serverless di DocumentDB a un cluster Amazon DocumentDB, anche il cluster deve avere il set di parametri. `ServerlessV2ScalingConfiguration`

Il `ServerlessV2ScalingConfiguration` parametro è composto da due valori che definiscono l'intervallo di capacità di scalabilità serverless di qualsiasi istanza serverless nel cluster:

- **`MinCapacity`**— La capacità di scalabilità minima di qualsiasi istanza serverless di DocumentDB nel cluster.
- **`MaxCapacity`**— La capacità di scalabilità massima di qualsiasi istanza serverless di DocumentDB nel cluster.

Scelta dell'**`MinCapacity`**impostazione per un cluster serverless DocumentDB

Si è tentati di scegliere sempre 0.5 per. `MinCapacity` Questo valore consente all'istanza di ridursi alla capacità minima quando è completamente inattiva, pur rimanendo attiva. Tuttavia, a seconda del modo in cui si utilizza il cluster e delle altre impostazioni configurate, una capacità minima diversa potrebbe essere la più efficace. Considera i seguenti fattori nella scelta dell'impostazione della capacità minima:

- La velocità di scalabilità di un'istanza serverless di DocumentDB dipende dalla sua capacità attuale. Maggiore è la capacità attuale, più veloce sarà il relativo dimensionamento verso l'alto. Se hai bisogno che l'istanza si scali rapidamente fino a una capacità molto elevata, valuta la possibilità di impostare la capacità minima su un valore in cui la velocità di scalabilità soddisfi i tuoi requisiti.
- Se in genere modifichi la classe di istanza delle tue istanze in previsione di un carico di lavoro particolarmente elevato o basso, puoi sfruttare questa esperienza per fare una stima approssimativa dell'intervallo di capacità serverless equivalente di DocumentDB. Per determinare la dimensione della memoria di un tipo di istanza Amazon DocumentDB su cui è stato effettuato il provisioning, consulta. Quote delle istanze

Ad esempio, supponiamo di utilizzare la classe di `db.r6g.xlarge` istanza quando il cluster ha un carico di lavoro basso. Quella classe di istanza ha 32 GiB di memoria. Pertanto, puoi specificare un valore compreso `MinCapacity` tra 16 per configurare un'istanza serverless che può essere ridimensionata all'incirca fino a raggiungere la stessa capacità. Questo perché ogni DCU corrisponde a circa 2 GiB di memoria. È possibile specificare un valore leggermente inferiore per consentire all'istanza di ridimensionarsi ulteriormente nel caso in cui a volte l'`db.r6g.xlarge`istanza fosse sottoutilizzata.

- Se l'applicazione funziona in modo più efficiente quando le istanze hanno una certa quantità di dati nella cache del buffer, valuta la possibilità di specificare un'impostazione DCU minima in cui la memoria sia sufficientemente grande da contenere i dati a cui si accede di frequente. Altrimenti, alcuni dati vengono eliminati dalla cache del buffer quando le istanze serverless si riducono a una dimensione di memoria inferiore. Quindi, quando le istanze vengono ridimensionate, le informazioni vengono lette nuovamente nella cache del buffer nel tempo. Se la quantità di dati da I/O riportare nella buffer cache è notevole, potrebbe essere più efficace scegliere un valore DCU minimo più alto. Per ulteriori informazioni, consulta Dimensionamento delle istanze.
- Se le istanze serverless di DocumentDB vengono eseguite per la maggior parte del tempo con una capacità particolare, valuta la possibilità di specificare un'impostazione di capacità minima inferiore a quella di base, ma non troppo inferiore. Le istanze serverless possono stimare in modo più efficace la quantità e la velocità di scalabilità quando la capacità attuale non è drasticamente inferiore alla capacità richiesta.
- Se il carico di lavoro su cui è stato eseguito il provisioning ha requisiti di memoria troppo elevati per classi di istanze di piccole dimensioni come T3 o T4g, scegli un'impostazione DCU minima che fornisca una memoria paragonabile a un'istanza R5 o R6g.
- Utilizza il seguente valore minimo `MinCapacity` per le funzionalità specificate (soggetto a modifiche):
  - Approfondimenti sulle prestazioni: 2 DCU
  - Cluster globali: 8 DCU (si applica solo al primario) Regione AWS
- In Amazon DocumentDB, la replica avviene a livello di storage, quindi la capacità del lettore non influisce direttamente sulla replica. Tuttavia, per le istanze di lettura serverless di DocumentDB che si scalano in modo indipendente, assicurati che la capacità minima sia sufficiente per gestire i carichi di lavoro durante i periodi di scrittura intensivi per evitare la latenza delle query. Se le istanze di lettura nei livelli di promozione 2—15 presentano problemi di prestazioni, valuta la possibilità di aumentare la capacità minima del cluster. Per dettagli su come modificare

la scalabilità delle istanze di lettura insieme a chi scrive o indipendentemente, consulta. Visualizzazione e modifica del livello di promozione dei lettori serverless

Se disponi di un cluster con istanze di lettura serverless di DocumentDB, i lettori non vengono scalati insieme all'istanza writer quando il livello di promozione dei lettori non è 0 o 1. In tal caso, l'impostazione di una capacità minima ridotta può comportare un eccessivo ritardo nella replica. Ciò è dovuto al fatto che le istanze di lettura potrebbero non avere capacità sufficiente per applicare le modifiche dell'istanza di scrittura quando il database è occupato. Imposta la capacità minima su un valore che rappresenti una quantità di memoria e CPU paragonabile a quella dell'istanza writer.

- Il tempo impiegato da un'istanza serverless di DocumentDB per passare dalla capacità minima alla capacità massima dipende dalla differenza tra i valori DCU minimi e massimi. Quando la capacità attuale dell'istanza è elevata, DocumentDB serverless aumenta con incrementi maggiori rispetto a quando l'istanza parte da una capacità ridotta. Pertanto, se si specifica una capacità massima relativamente elevata e l'istanza trascorre la maggior parte del tempo vicino a tale capacità, è consigliabile aumentare l'impostazione minima della DCU. In questo modo, un'istanza inattiva può tornare alla capacità massima più rapidamente.
- Alcuni limiti di istanza sono determinati dalla capacità attuale dell'istanza serverless, ad esempio il limite di connessioni, il limite del cursore e il limite delle transazioni aperte. Se la capacità attuale dell'istanza è ridotta, anche i limiti saranno anch'essi ridotti. Se questi limiti rappresentano un problema quando l'istanza serverless viene ridotta al suo `MinCapacity` valore, valuta la possibilità di `MinCapacity` aumentarli a un valore più alto. Per ulteriori informazioni, consulta Limiti delle istanze serverless di Amazon DocumentDB.
- Inoltre, alcuni limiti di istanze sono limitati a un valore massimo inferiore se impostato su un valore inferiore o uguale a 1,0 DCU, ad esempio il limite delle connessioni attive, il limite del cursore e il limite delle transazioni aperte. `MinCapacity` Se questi limiti massimi non sono sufficienti per il carico di lavoro, utilizza un `MinCapacity` valore di almeno 1,5 DCU. Per ulteriori informazioni, consulta Limiti delle istanze serverless di Amazon DocumentDB.

Per istruzioni su come modificare la configurazione di scalabilità di un cluster, consulta. Gestione di Amazon DocumentDB senza server

Scelta dell'**`MaxCapacity`**impostazione per un cluster serverless DocumentDB

Si è tentati di scegliere sempre un valore elevato per l'impostazione massima della capacità serverless di DocumentDB. Una capacità massima elevata consente all'istanza di scalare al massimo

quando esegue un carico di lavoro intensivo. Un valore basso evita il rischio di dover sostenere di addebiti imprevisti. A seconda di come si utilizza il cluster e delle altre impostazioni configurate, il valore più efficace potrebbe essere maggiore o minore basso rispetto a quello originariamente individuato. Considera i seguenti fattori nella scelta dell'impostazione della capacità massima:

- La capacità massima deve essere almeno pari alla capacità minima. Puoi impostare lo stesso valore per la capacità minima e la capacità massima. Tuttavia, in questo caso la capacità non aumenta né diminuisce mai. Pertanto, a parte gli scenari di test, l'uso di valori identici per la capacità minima e massima non è appropriato.
- La capacità massima deve essere di almeno 1,0 DCU e deve essere al massimo di 256 DCU.
- Monitora la scalabilità e l'utilizzo delle risorse delle tue istanze serverless. Se la tua istanza serverless viene spesso scalata fino alla capacità massima e soddisfa i vincoli di risorse (ad esempio, quando la `DCUUtilization` metrica è pari a 100,0), seleziona un valore più alto. `MaxCapacity` Per ulteriori informazioni, consulta Monitoraggio di Amazon DocumentDB senza server.
- Se in genere modifichi la classe di istanze delle istanze assegnate in previsione di un carico di lavoro particolarmente elevato o basso, puoi utilizzare tale esperienza per stimare l'intervallo di capacità serverless equivalente di DocumentDB. Per determinare la dimensione della memoria delle istanze Amazon DocumentDB sottoposte a provisioning, consulta. Quote delle istanze

  Ad esempio, supponiamo di utilizzare la classe di `db.r6g.4xlarge` istanza quando il cluster ha un carico di lavoro elevato. Quella classe di istanza ha 128 GiB di memoria. Pertanto, è possibile specificare un'impostazione DCU massima di 64 per configurare un'istanza serverless in grado di scalare fino a circa la stessa capacità. Questo perché ogni DCU corrisponde a circa 2 GiB di memoria. È possibile specificare un valore leggermente più alto per consentire all'istanza di scalare ulteriormente nel caso in cui a volte l'`db.r6g.4xlarge`istanza non abbia una capacità sufficiente per gestire il carico di lavoro in modo efficace.
- Se hai un limite di budget per l'utilizzo del database, scegli un valore che rimanga entro tale limite anche se tutte le istanze serverless funzionano sempre alla massima capacità. Ricorda che quando hai n istanze serverless nel tuo cluster, la capacità serverless massima teorica che il cluster può consumare in qualsiasi momento è n volte l'impostazione DCU massima per il cluster. La quantità effettiva consumata potrebbe essere inferiore, ad esempio se alcune istanze di lettura vengono dimensionate in modo indipendente rispetto all'istanza di scrittura.
- Se utilizzi istanze serverless reader per scaricare parte del carico di lavoro di sola lettura dall'istanza writer, potresti scegliere un'impostazione di capacità massima inferiore. Lo fai per

capire che non è necessario scalare ogni istanza del lettore come se il cluster contenesse solo una singola istanza.

- Supponiamo di voler evitare un uso eccessivo a causa di parametri del database configurati in modo errato o di query inefficienti nell'applicazione. In tal caso, è possibile evitare un uso eccessivo accidentale scegliendo un'impostazione di capacità massima inferiore a quella più alta possibile.
- Se i picchi dovuti alla reale attività degli utenti sono rari ma si verificano, è possibile tenerne conto quando si sceglie l'impostazione della capacità massima. Se è prioritario che l'applicazione continui a funzionare con prestazioni e scalabilità complete, è possibile specificare un'impostazione di capacità massima superiore a quella osservata durante il normale utilizzo. Se l'applicazione può essere eseguita con una capacità di throughput effettiva ridotta durante picchi di attività molto estremi, è possibile scegliere un'impostazione di capacità massima leggermente inferiore. Assicurati di scegliere un'impostazione che disponga ancora di memoria e risorse CPU sufficienti per mantenere l'applicazione in esecuzione.
- Se nel cluster attivate le impostazioni che aumentano l'utilizzo della memoria per ogni istanza, tenete conto di tale memoria quando decidete il valore massimo della DCU. Tali impostazioni includono quelle per Performance Insights e i cluster globali. Assicurati che il valore DCU massimo consenta alle istanze serverless di scalare abbastanza da gestire il carico di lavoro quando tali funzionalità vengono utilizzate. Per informazioni sulla risoluzione dei problemi causati dalla combinazione di un'impostazione DCU massima bassa e delle funzionalità di Amazon DocumentDB che impongono un sovraccarico di memoria, consulta (di seguito). Esclusione di errori dovuti all'esaurimento della memoria
- Utilizza il seguente valore minimo `MaxCapacity` per le funzionalità specificate (soggetto a modifiche):
  - Creazione di istanze serverless su un cluster con un grande volume di dati: 2 DCU (inclusa la creazione di istanze serverless come parte di un ripristino del cluster).
- Alcuni limiti di istanza sono determinati dalla capacità attuale dell'istanza, ad esempio il limite di connessioni, il limite del cursore e il limite delle transazioni aperte. Quando scegli il `MaxCapacity` valore per il tuo carico di lavoro, assicurati di tenere a mente questi limiti di istanze per evitare di trovarti bloccato da uno di questi limiti. Per ulteriori informazioni, consulta Limiti delle istanze serverless di Amazon DocumentDB.

Per istruzioni su come modificare la configurazione di scalabilità di un cluster, consulta. Gestione di Amazon DocumentDB senza server

Esclusione di errori dovuti all'esaurimento della memoria

Se una delle tue istanze serverless di DocumentDB raggiunge costantemente il limite della sua capacità massima, Amazon DocumentDB indica questa condizione impostando l'istanza su uno stato di parametri incompatibili. Sebbene l'istanza abbia lo stato di parametri incompatibili, alcune operazioni sono bloccate. Ad esempio, non è possibile aggiornare la versione del motore. Per ulteriori informazioni sullo stato di un'istanza Amazon DocumentDB, consulta. Monitoraggio dello stato di un'istanza Amazon DocumentDB

In genere, l'istanza passa a questo stato quando viene riavviata frequentemente a causa di errori di esaurimento della memoria. Amazon DocumentDB registra un evento quando si verifica questo tipo di riavvio. Per visualizzare gli eventi delle risorse, consultaVisualizzazione degli eventi di Amazon DocumentDB. L'utilizzo insolitamente elevato della memoria può verificarsi a causa del sovraccarico derivante dall'attivazione di impostazioni come Performance Insights. Può anche derivare da un carico di lavoro elevato sull'istanza o dalla gestione dei metadati associati a un numero elevato di oggetti dello schema.

Se la pressione della memoria diminuisce e l'istanza non raggiunge la sua capacità massima molto spesso, Amazon DocumentDB riporta automaticamente lo stato dell'istanza in disponibile.

Per gestire in modo ottimale questa situazione, è possibile eseguire alcune o tutte le seguenti azioni:

- Aumenta il limite inferiore di capacità per le istanze serverless modificando il valore minimo dell'unità di capacità (DCU) di DocumentDB per il cluster. In questo modo si evitano i problemi in cui un database inattivo viene ridotto a una capacità di memoria inferiore a quella necessaria per le funzionalità attivate nel cluster. Dopo aver modificato le impostazioni DCU per il cluster, riavvia l'istanza serverless. In questo modo si valuta se Amazon DocumentDB è in grado di ripristinare lo stato su disponibile.
- Aumenta il limite massimo di capacità per le istanze serverless modificando il valore DCU massimo per il cluster. Ciò consente di evitare problemi nel caso in cui un database occupato non sia in grado di eseguire il dimensionamento verso l'alto fino a una capacità con memoria sufficiente per le funzionalità attivate nel cluster e nel carico di lavoro del database. Dopo aver modificato le impostazioni DCU per il cluster, riavvia l'istanza serverless. In questo modo si valuta se Amazon DocumentDB è in grado di ripristinare lo stato su disponibile.
- Disattivare le impostazioni di configurazione che richiedono un sovraccarico di memoria. Ad esempio, supponiamo di avere una funzionalità come Performance Insights attivata ma di non utilizzarla. In tal caso, puoi disattivarla. Oppure puoi aumentare i valori di capacità minima e massima del cluster per tenere conto della memoria utilizzata da questi tipi di funzionalità. Per

le linee guida sulla scelta delle impostazioni di capacità minima e massima, consulta Scelta dell'intervallo di capacità di scalabilità per un cluster serverless DocumentDB.

- Riduci il carico di lavoro sull'istanza. Ad esempio, puoi aggiungere istanze di lettura al cluster per distribuire il carico delle query di sola lettura su più istanze.

Perché la mia istanza serverless non viene ridimensionata?

In alcuni casi, DocumentDB serverless non si riduce alla capacità minima, anche senza carico sul database. Questo può accadere per i seguenti motivi:

- Performance Insights può aumentare l'utilizzo delle risorse e impedire che il database si riduca alla capacità minima. Le caratteristiche principali comprendono:
- Se un'istanza di lettura non viene ridotta al minimo e mantiene una capacità uguale o superiore a quella dell'istanza writer, controlla il livello di priorità dell'istanza del lettore. Le istanze di lettura serverless di DocumentDB di livello 0 o 1 sono mantenute a una capacità minima almeno pari a quella dell'istanza writer. Imposta il livello di priorità dell'istanza di lettura almeno su 2 in modo che possa essere aumentata o ridotta indipendentemente dall'istanza di scrittura. Per ulteriori informazioni, consulta Scalabilità senza server di Amazon DocumentDB.
- I carichi di lavoro di database gravosi possono aumentare l'utilizzo delle risorse.
- I volumi di database di grandi dimensioni possono aumentare l'utilizzo delle risorse. Amazon DocumentDB utilizza risorse di memoria e CPU per la gestione dei cluster. Amazon DocumentDB richiede più CPU e memoria per gestire cluster con volumi di database più grandi. Se la capacità minima del cluster è inferiore a quella minima richiesta per la gestione dei cluster, il cluster non verrà ridotto fino alla capacità minima.
- L'attività di manutenzione in background può aumentare periodicamente l'utilizzo delle risorse.

Se il database continua a non essere ridimensionato fino alla capacità minima configurata, arrestalo e riavvialo per recuperare eventuali frammenti di memoria che potrebbero essersi accumulati nel tempo.

> **Note**
>
> L'arresto e l'avvio di un database comportano tempi di inattività. Fatelo solo se necessario.

Monitoraggio di Amazon DocumentDB senza server

Per ulteriori informazioni sul monitoraggio in Amazon DocumentDB, consulta. Monitoraggio e registrazione in Amazon DocumentDB

Argomenti

- Memoria insufficiente: stato dei parametri incompatibili
- Metriche Amazon per DocumentDB serverless CloudWatch
- Monitoraggio delle prestazioni serverless di DocumentDB con Performance Insights

Memoria insufficiente: stato dei parametri incompatibili

Se una delle tue istanze serverless raggiunge costantemente il limite della sua capacità massima, Amazon DocumentDB indica questa condizione impostando l'istanza su uno stato di parametri incompatibili. Per ulteriori informazioni, consulta Esclusione di errori dovuti all'esaurimento della memoria.

Metriche Amazon per DocumentDB serverless CloudWatch

Per ulteriori informazioni sull'utilizzo CloudWatch con Amazon DocumentDB, consulta. Monitoraggio di Amazon DocumentDB con CloudWatch

Puoi visualizzare le tue istanze serverless in CloudWatch per monitorare la capacità consumata da ciascuna istanza con la metrica. `ServerlessDatabaseCapacity` Puoi anche monitorare tutte le CloudWatch metriche standard di DocumentDB, ad esempio e Queries. `DatabaseConnections` Per l'elenco completo delle CloudWatch metriche che puoi monitorare per Amazon DocumentDB, consulta. Monitoraggio di Amazon DocumentDB con CloudWatch Le seguenti metriche a CloudWatch livello di istanza sono importanti da monitorare per capire in che modo le istanze serverless DocumentDB si scalano verso l'alto e verso il basso. Tutti questi parametri vengono calcolati ogni secondo. In questo modo, puoi monitorare lo stato attuale delle tue istanze serverless. Puoi impostare allarmi per notificarti se un'istanza serverless si avvicina a una soglia per le metriche relative alla capacità. È possibile determinare se le impostazioni di capacità minima e massima sono appropriate o se è necessario adeguarle. È possibile individuare le aree da valutare con maggiore attenzione per ottimizzare l'efficienza del database.

- **`ServerlessDatabaseCapacity`**— Come metrica a livello di istanza, riporta il numero di DCU rappresentati dalla capacità corrente dell'istanza. Come metrica a livello di cluster, rappresenta la

media dei `ServerlessDatabaseCapacity` valori di tutte le istanze serverless di DocumentDB nel cluster.

- **`DCUUtilization`.**— Questa metrica è nuova in DocumentDB serverless. Questo valore è rappresentato come percentuale. Viene calcolato come il valore della `ServerlessDatabaseCapacity` metrica diviso per il valore DCU massimo del cluster. Considera le seguenti linee guida per l'interpretazione di questo parametro e per valutare quale azione eseguire:
  - Se questa metrica si avvicina a un valore di`100.0`, l'istanza è aumentata il più possibile. Valuta la possibilità di aumentare l'impostazione massima della DCU per il cluster. In questo modo, sia le istanze di scrittura che quelle di lettura possono raggiungere una capacità superiore.
  - Supponiamo che un carico di lavoro di sola lettura faccia sì che un'istanza reader si avvicini a un `DCUUtilization` of`100.0`, mentre l'istanza writer non sia vicina alla sua capacità massima. In questo caso, valuta la possibilità di aggiungere ulteriori istanze di lettura al cluster. In questo modo, puoi distribuire la parte di sola lettura del carico di lavoro su più istanze, riducendo il carico su ciascuna istanza di lettore.
  - Supponiamo di eseguire un'applicazione di produzione, in cui prestazioni e scalabilità sono fattori di primaria importanza. In questo caso, puoi impostare il valore DCU massimo per il cluster su un numero elevato. Il tuo obiettivo è che la `DCUUtilization` metrica sia sempre inferiore a 100,0. Con un valore DCU massimo elevato, puoi essere certo che c'è abbastanza spazio in caso di picchi imprevisti nell'attività del database. Dovrai sostenere solo i costi relativi al consumo effettivo di capacità del database.
- **`CPUUtilization`**— Questa metrica viene interpretata in modo diverso in DocumentDB serverless rispetto alle istanze con provisioning. Per DocumentDB serverless, questo valore è una percentuale calcolata come la quantità di CPU attualmente utilizzata, divisa per la capacità della CPU disponibile con il valore massimo DCU del cluster. Amazon DocumentDB monitora questo valore automaticamente e aumenta la scalabilità dell'istanza serverless quando l'istanza utilizza costantemente una percentuale elevata della sua capacità di CPU.

  Se questa metrica si avvicina a un valore di`100.0`, l'istanza ha raggiunto la capacità massima della CPU. Valuta la possibilità di aumentare l'impostazione massima della DCU per il cluster. Se questa metrica si avvicina al valore di `100.0` un'istanza di lettore, valuta la possibilità di aggiungere ulteriori istanze di lettore al cluster. In questo modo, puoi distribuire la parte di sola lettura del carico di lavoro su più istanze, riducendo il carico su ciascuna istanza di lettore.
- **`FreeableMemory`**— Questo valore rappresenta la quantità di memoria inutilizzata disponibile quando l'istanza serverless DocumentDB viene scalata alla sua capacità massima. Per ogni DCU

la cui capacità attuale è inferiore alla capacità massima, questo valore aumenta di circa 2 GiB. Pertanto, questa metrica non si avvicina allo zero finché l'istanza non viene scalata al massimo.

Se questa metrica si avvicina a un valore pari a zero, l'istanza è aumentata il più possibile e si sta avvicinando al limite della memoria disponibile. Valuta la possibilità di aumentare l'impostazione massima della DCU per il cluster. Se questa metrica si avvicina a un valore pari a zero su un'istanza del lettore, valuta la possibilità di aggiungere ulteriori istanze di lettore al cluster. In questo modo, la parte di sola lettura del carico di lavoro può essere distribuita su più istanze, riducendo l'utilizzo della memoria su ciascuna istanza del lettore.

- **`TempStorageIops`**— Il numero di IOPS eseguiti sullo storage locale collegato all'istanza. Include gli IOPS per le operazioni sia di letture che di scrittura. Questo parametro rappresenta un conteggio e viene misurato una volta al secondo. Questa è una nuova metrica per DocumentDB serverless. Per informazioni dettagliate, vedi Monitoraggio di Amazon DocumentDB con CloudWatch.
- **`TempStorageThroughput`**— La quantità di dati trasferiti da e verso l'archiviazione locale associata all'istanza. Questo parametro rappresenta i byte e viene misurato una volta al secondo. Questa è una nuova metrica per DocumentDB serverless. Per informazioni dettagliate, vedi Monitoraggio di Amazon DocumentDB con CloudWatch.

In genere, la maggior parte della scalabilità per le istanze serverless di DocumentDB è causata dall'utilizzo della memoria e dall'attività della CPU. Le `TempStorageThroughput` metriche `TempStorageIops` e possono aiutarti a diagnosticare i rari casi in cui l'attività di rete per i trasferimenti tra l'istanza e i dispositivi di archiviazione locali è responsabile di aumenti imprevisti della capacità. Per monitorare altre attività di rete, puoi utilizzare i seguenti parametri:

- `NetworkReceiveThroughput`
- `NetworkThroughput`
- `NetworkTransmitThroughput`
- `StorageNetworkReceiveThroughput`
- `StorageNetworkThroughput`
- `StorageNetworkTransmitThroughput`

In che modo le CloudWatch metriche serverless di DocumentDB si applicano al tuo AWS fattura

Gli addebiti serverless di DocumentDB in AWS fattura vengono calcolati in base alla stessa `ServerlessDatabaseCapacity` metrica che puoi monitorare. Il meccanismo di fatturazione può differire dalla CloudWatch media calcolata di questa metrica nei casi in cui si utilizza la capacità serverless di DocumentDB solo per una parte di un'ora. Può inoltre differire se problemi di sistema rendono la CloudWatch metrica non disponibile per brevi periodi. Pertanto, potresti visualizzare un valore leggermente diverso DCU-hours sulla bolletta rispetto a quando calcolassi tu stesso il numero dal valore `ServerlessDatabaseCapacity` medio.

Esempi di comandi Amazon CloudWatch CLI per le metriche serverless di DocumentDB

I seguenti AWS CLI esempi dimostrano come è possibile monitorare le CloudWatch metriche più importanti relative a DocumentDB serverless. In ogni caso, sostituite la `Value=` stringa del `--dimensions` parametro con l'identificatore della vostra istanza serverless di DocumentDB.

Il seguente esempio di Linux mostra i valori di capacità minima, massima e media per un'istanza, misurati ogni 10 minuti nell'arco di un'ora. I comandi di data di Linux specificano le ore di inizio e di fine relative alla data e all'ora correnti. La funzione `sort_by` nel parametro `--query` ordina i risultati cronologicamente in base al campo `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
```

Il seguente esempio di Linux dimostra il monitoraggio della capacità di un'istanza in un cluster. Misura l'utilizzo minimo, massimo e medio della capacità di un'istanza. Le misurazioni vengono effettuate una volta ogni ora per un periodo di tre ore. Questi esempi utilizzano la `DCUUtilization` metrica che rappresenta una percentuale del limite massimo delle DCU, anziché

`ServerlessDatabaseCapacity` rappresentare un numero fisso di DCU. In questo modo, non è necessario conoscere i numeri effettivi relativi ai valori DCU minimo e massimo nell'intervallo di capacità. Puoi visualizzare percentuali che vanno da 0 a 100.

```
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
```

Il seguente esempio Linux esegue misurazioni simili a quelle precedenti. In questo caso, le misure fanno riferimento al parametro `CPUUtilization`. Le misurazioni vengono effettuate ogni 10 minuti per un periodo di 1 ora. I numeri rappresentano la percentuale di CPU disponibile utilizzata, in base alle risorse CPU disponibili per l'impostazione della capacità massima per l'istanza.

```
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
```

Il seguente esempio Linux esegue misurazioni simili a quelle precedenti. In questo caso, le misure fanno riferimento al parametro `FreeableMemory`. Le misurazioni vengono effettuate ogni 10 minuti per un periodo di 1 ora.

```
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
```

Monitoraggio delle prestazioni serverless di DocumentDB con Performance Insights

È possibile utilizzare Performance Insights per monitorare le prestazioni delle istanze serverless di DocumentDB. Per le procedure di Approfondimenti sulle prestazioni, consulta Monitoraggio con Performance Insights.

I seguenti nuovi contatori Performance Insights si applicano alle istanze serverless di DocumentDB:

- **`os.general.serverlessDBCapacity`**— La capacità attuale dell'istanza nei DCU. Il valore corrisponde alla `ServerlessDatabaseCapacity` CloudWatch metrica dell'istanza.
- **`os.general.dcuUtilization`**— La percentuale della capacità attuale rispetto alla capacità massima configurata. Il valore corrisponde alla `DCUUtilization` CloudWatch metrica dell'istanza.
- **`os.general.maxConfiguredDcu`**— La capacità massima configurata per questa istanza serverless di DocumentDB. Viene misurata in DCU.
- **`os.general.minConfiguredDcu`**— La capacità minima configurata per questa istanza serverless di DocumentDB. Viene misurata in DCU.

Per l'elenco completo dei contatori di Approfondimenti sulle prestazioni, consulta Performance Insights per le contrometriche.

Quando i valori di vCPU vengono visualizzati per un'istanza serverless di DocumentDB in Performance Insights, tali valori rappresentano stime basate sul valore DCU per l'istanza. Per intervalli di default di un minuto, tutti i valori vCPU frazionari vengono arrotondati al numero intero più vicino. Per intervalli più lunghi, il valore vCPU visualizzato è la media dei valori vCPU interi per ogni minuto.

Utilizzo di Amazon DocumentDB Elastic Clusters

I cluster elastici Amazon DocumentDB supportano carichi di lavoro con milioni di reads/writes petabyte di capacità di storage al secondo. I cluster elastici semplificano inoltre il modo in cui gli sviluppatori interagiscono con Amazon DocumentDB eliminando la necessità di scegliere, gestire o aggiornare le istanze.

> **Note**
>
> I cluster elastici utilizzano un protocollo cablato compatibile con MongoDB 5.0. Non sono supportati nella versione 8.0 del motore Amazon DocumentDB e non sono disponibili nelle regioni della Cina o. AWS GovCloud

I cluster elastici Amazon DocumentDB sono stati creati per:

- Fornisci una soluzione per i clienti che cercano un database che offra una scalabilità praticamente illimitata con ricche funzionalità di interrogazione e compatibilità con le API MongoDB.
- Offri ai clienti limiti di connessione più elevati e riduci i tempi di inattività dovuti all'applicazione delle patch.
- Continua a investire in un'architettura nativa per il cloud, elastica e all'avanguardia per i carichi di lavoro JSON.

Argomenti

- Casi d'uso di Elastic Cluster
- Vantaggi dei cluster elastici
- Disponibilità della regione e della versione di Elastic Cluster
- Limitazioni
- Cluster elastici Amazon DocumentDB: come funzionano
- Inizia a usare i cluster elastici Amazon DocumentDB
- Best practice per i cluster elastici Amazon DocumentDB
- Gestione dei cluster elastici Amazon DocumentDB
- Crittografia dei dati inattivi per i cluster elastici Amazon DocumentDB

- Service-linked ruoli nei cluster elastici

Casi d'uso di Elastic Cluster

I database di documenti sono utili per i carichi di lavoro che richiedono uno schema flessibile per uno sviluppo rapido e iterativo. Ad esempio, i casi d'uso di Amazon DocumentDB, consultaCasi d'uso del database di documenti.

Di seguito sono riportati alcuni esempi di casi d'uso per i quali i cluster elastici possono offrire vantaggi significativi:

Profili utente

Poiché i database di documenti hanno uno schema flessibile, possono archiviare documenti con attributi e valori di dati diversi su larga scala. I cluster elastici sono una soluzione pratica ai profili online in cui diversi utenti forniscono diversi tipi di informazioni. Supponiamo che le tue applicazioni supportino centinaia di milioni di profili utente. È possibile utilizzare i cluster elastici per supportare tali applicazioni perché possono essere scalate verso l'alto e verso l'esterno per supportare milioni di scritture e letture su questi profili utente. Puoi anche ridimensionare le ore non di punta per ridurre i costi.

Gestione dei contenuti e record storici

Per gestire in modo efficiente i contenuti, devi poterli raccogliere e aggregare da diverse origini e quindi inviarli al cliente. Grazie al loro schema flessibile, i database di documenti sono perfetti per raccogliere e archiviare qualsiasi tipo di dati. Puoi utilizzarli per creare e incorporare nuovi tipi di contenuti, inclusi contenuti generati dagli utenti come immagini, commenti e video. Nel tempo, il database potrebbe richiedere più spazio di archiviazione. Con i cluster elastici, puoi distribuire i tuoi dati su più volumi di archiviazione, consentendoti di archiviare petabyte di dati in un singolo cluster.

Vantaggi dei cluster elastici

AWS integrazione dei servizi

I cluster elastici di Amazon DocumentDB si integrano con altri AWS servizi nello stesso modo in cui Amazon DocumentDB fa:

- Migrazione: puoi utilizzare AWS Database Migration Service (DMS) per migrare da MongoDB e altri database relazionali ai cluster elastici di Amazon DocumentDB.
- Monitoraggio: puoi monitorare lo stato e le prestazioni del tuo cluster elastico utilizzando Amazon. CloudWatch
- Sicurezza: puoi configurare l'autenticazione e l'autorizzazione tramite AWS Identity and Access Management (IAM) per gestire le risorse del tuo cluster elastico, ad esempio la creazione e l'aggiornamento di un cluster elastico, e utilizzare Amazon VPC per VPC-only connessioni sicure. L'accesso con autenticazione IAM per i cluster elastici non è supportato.
- Gestione dei dati: puoi utilizzare AWS Glue per importare ed esportare dati from/to altri AWS servizi come Amazon S3, Amazon Redshift e Amazon Service. OpenSearch

Disponibilità della regione e della versione di Elastic Cluster

Disponibilità delle regioni per i cluster elastici

La tabella seguente mostra Regioni AWS dove sono attualmente disponibili i cluster elastici Amazon DocumentDB e l'endpoint per ciascuna regione.

Nome Regione	Regione	Zone di disponibilità
Stati Uniti orientali (Virginia settentrionale)	`us-east-1`	5
Stati Uniti orientali (Ohio)	`us-east-2`	3
Stati Uniti occidentali (Oregon)	`us-west-2`	3
Asia Pacifico (Hong Kong)	`ap-east-1`	3
Asia Pacifico (Mumbai)	`ap-south-1`	3
Asia Pacifico (Seul)	`ap-northeast-2`	3
Asia Pacifico (Singapore)	`ap-southeast-1`	3
Asia Pacifico (Sydney)	`ap-southeast-2`	3
Asia Pacifico (Tokyo)	`ap-northeast-1`	3

Nome Regione	Regione	Zone di disponibilità
Canada (Centrale)	`ca-central-1`	3
Sud America (San Paolo)	`sa-east-1`	3
Europa (Francoforte)	`eu-central-1`	3
Europa (Irlanda)	`eu-west-1`	3
Europa (Londra)	`eu-west-2`	3
Europa (Milano)	`eu-south-1`	3
Europa (Parigi)	`eu-west-3`	3

Disponibilità delle versioni

I cluster elastici supportano il protocollo wire compatibile con MongoDB 5.0. Per le differenze tra i cluster basati su istanze Amazon DocumentDB 4.0 e i cluster elastici, consulta. Differenze funzionali tra Amazon DocumentDB 4.0 e i cluster elastici Nota: i cluster elastici non sono supportati su Amazon DocumentDB 8.0

Limitazioni

Gestione elastica dei cluster

Le seguenti funzionalità e funzionalità di gestione dei cluster non sono supportate in questa versione:

- Porta personalizzata (è supportata solo la 27017)
- Possibilità di utilizzare utenti e ruoli IAM per autenticarsi nel database
- Capacità di creare cluster globali
- Eventi Amazon DocumentDB esistenti e iscrizione agli eventi
- Partizionamento dell'intervallo
- Condividi una raccolta esistente

- Multi-field chiave condivisa
- Cambia la chiave shard
- Point-in-time ripristinare
- Clonazione
- Approfondimenti sulle prestazioni

> **Note**
>
> Per informazioni sulle quote dei cluster elastici, consultaQuote Amazon DocumentDB.

Operazioni di interrogazione e scrittura

I seguenti comandi e funzionalità per le operazioni di interrogazione e scrittura non sono supportati in questa versione:

- Comandi DDL durante le operazioni di ridimensionamento
- Profiler
- Gruppi di parametri
- AWS Config
- AWS Backup

Gestione della raccolta e dell'indice

Le seguenti funzionalità di gestione delle raccolte e degli indici non sono supportate in questa versione:

- Indici univoci
- Indici parziali
- Indici di testo
- Indici vettoriali
- Compressione dei documenti

Amministrazione e diagnostica

I seguenti comandi e funzionalità di amministrazione e diagnostica non sono supportati in questa versione:

- AWS Secrets Manager
- Role-based-access-control ruoli personalizzati (RBAC).
- Durante la connessione, il problema di scrittura pari a 0 non è supportato.
- Modifica delle sottoreti appartenenti a un VPC attualmente non assegnato a un cluster elastico esistente.

Opt-in features

Le seguenti funzionalità di attivazione di Amazon DocumentDB non sono supportate in questa versione:

- Transazioni ACID
- DDL/DML controllo
- Change streams
- Comandi di sessione

Cluster elastici Amazon DocumentDB: come funzionano

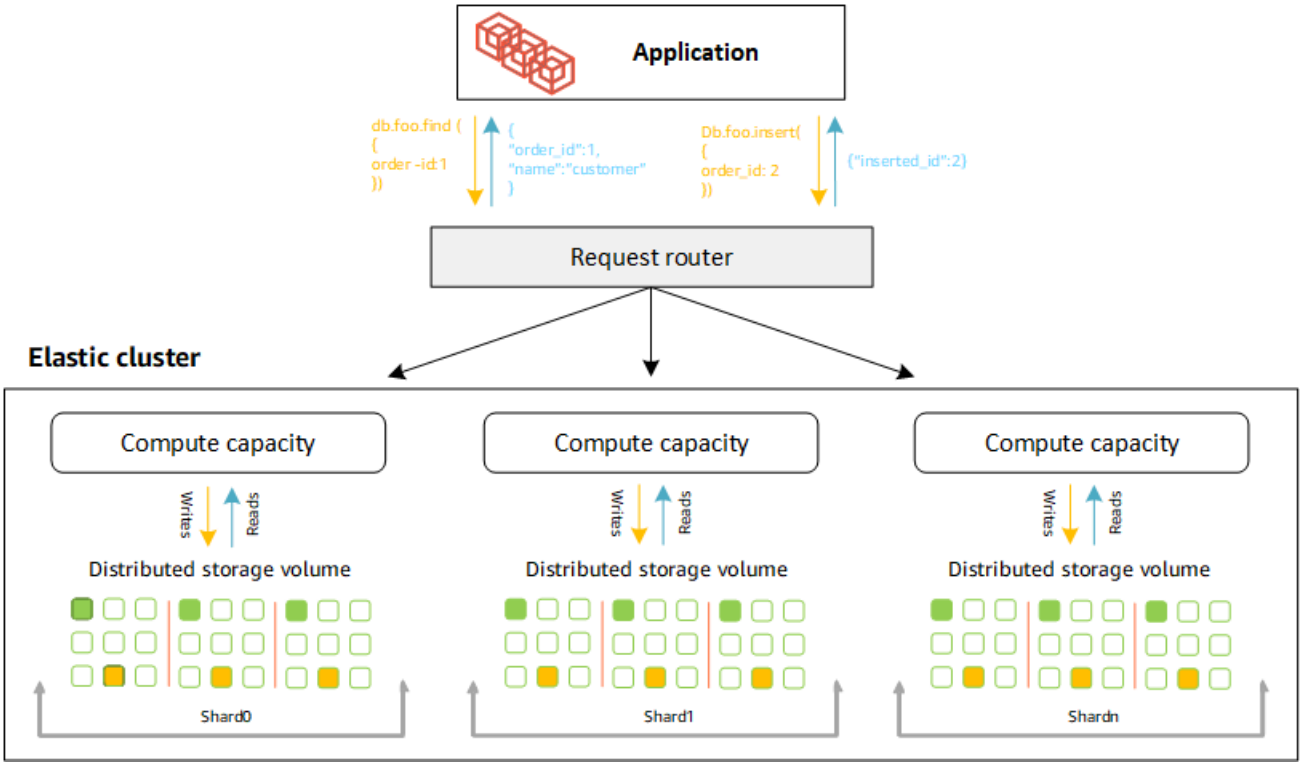
Gli argomenti di questa sezione forniscono informazioni sui meccanismi e le funzioni alla base dei cluster elastici di Amazon DocumentDB.

Argomenti

- Sharding elastico dei cluster Amazon DocumentDB
- Migrazione elastica dei cluster
- Scalabilità elastica dei cluster
- Affidabilità del cluster elastico
- Storage e disponibilità dei cluster elastici
- Differenze funzionali tra Amazon DocumentDB 4.0 e i cluster elastici

Sharding elastico dei cluster Amazon DocumentDB

I cluster elastici Amazon DocumentDB utilizzano lo sharding basato su hash per partizionare i dati in un sistema di storage distribuito. Lo sharding, noto anche come partizionamento, divide grandi set di dati in piccoli set di dati su più nodi, consentendoti di scalare il database oltre i limiti di scalabilità verticale. I cluster elastici utilizzano la separazione, o «disaccoppiamento», tra elaborazione e storage in Amazon DocumentDB, consentendoti di scalare indipendentemente l'uno dall'altro. Anziché ripartizionare le raccolte spostando piccoli blocchi di dati tra nodi di calcolo, i cluster elastici copiano i dati in modo efficiente all'interno del sistema di storage distribuito.



Definizioni di shard

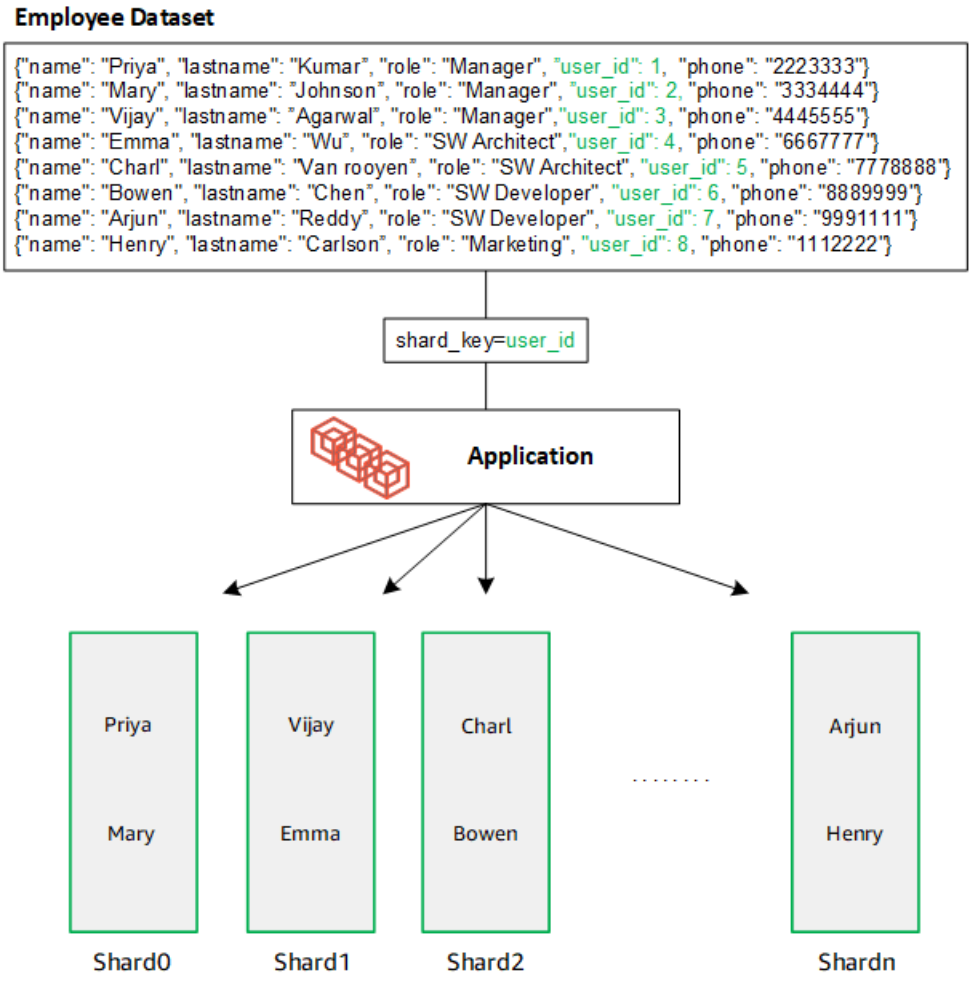
Definizioni della nomenclatura degli shard:

- Shard: uno shard fornisce il calcolo per un cluster elastico. Avrà una singola istanza di scrittore e da 0 a 15 repliche di lettura. Per impostazione predefinita, uno shard avrà due istanze: una replica di scrittura e una singola replica di lettura. È possibile configurare un massimo di 32 shard e ogni istanza dello shard può avere un massimo di 64 vCPU.
- Shard key: una shard key è un campo obbligatorio nei documenti JSON nelle raccolte frammentate che i cluster elastici utilizzano per distribuire il traffico di lettura e scrittura allo shard corrispondente.

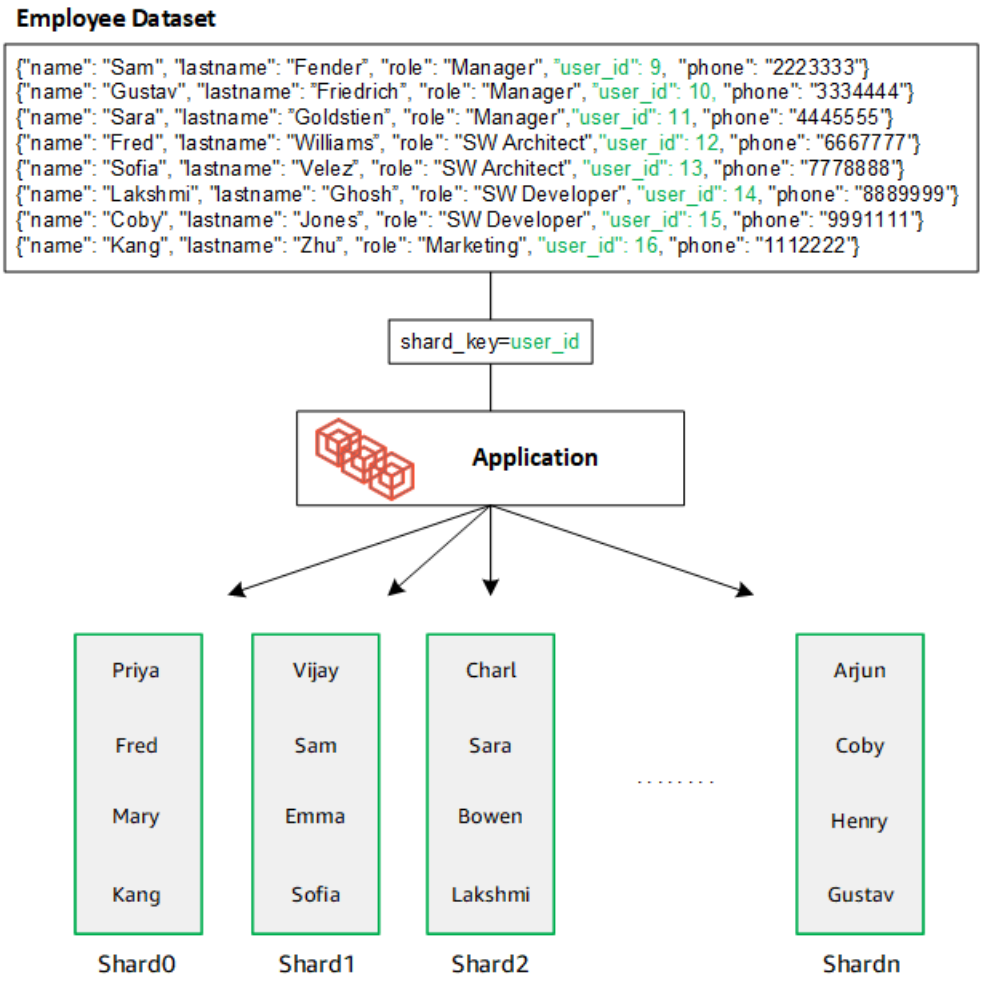
- Raccolta frammentata: una raccolta frammentata è una raccolta i cui dati sono distribuiti su un cluster elastico in partizioni di dati.
- Partizione: una partizione è una porzione logica di dati frammentati. Quando si crea una raccolta frammentata, i dati vengono organizzati automaticamente in partizioni all'interno di ogni frammento in base alla chiave dello shard. Ogni frammento ha più partizioni.

Distribuzione dei dati tra frammenti configurati

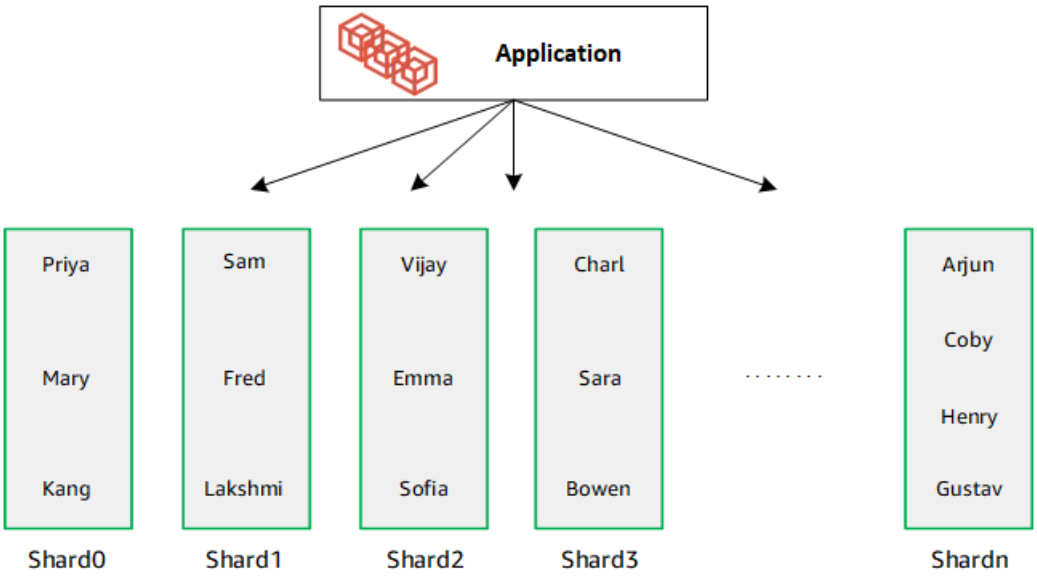
Crea una chiave shard con molti valori univoci. Una buona chiave shard ripartizionerà uniformemente i dati tra gli shard sottostanti, offrendo al carico di lavoro il throughput e le prestazioni migliori. L'esempio seguente sono i dati relativi ai nomi dei dipendenti che utilizzano una chiave shard denominata «user_id»:



DocumentDB utilizza l'hash sharding per suddividere i dati tra gli shard sottostanti. I dati aggiuntivi vengono inseriti e distribuiti nello stesso modo:



Quando si espande il database aggiungendo frammenti aggiuntivi, Amazon DocumentDB ridistribuisce automaticamente i dati:



Migrazione elastica dei cluster

Amazon DocumentDB supporta la migrazione dei dati frammentati di MongoDB verso cluster elastici. Sono supportati metodi di migrazione offline, online e ibridi. Per ulteriori informazioni, consulta Migrazione e aggiornamento di Amazon DocumentDB.

Scalabilità elastica dei cluster

I cluster elastici Amazon DocumentDB offrono la possibilità di aumentare il numero di shard (scalabilità orizzontale) nel cluster elastico e il numero di vCPU applicate a ciascun shard (scalabilità verso l'alto). Puoi anche ridurre il numero di shard e la capacità di calcolo (vCPU) secondo necessità.

Per le best practice di scalabilità, consulta. Scalabilità dei cluster elastici

> **Note**
>
> Cluster-level è disponibile anche il ridimensionamento. Per ulteriori informazioni, consulta Scalabilità dei cluster Amazon DocumentDB.

Affidabilità del cluster elastico

Amazon DocumentDB è progettato per essere affidabile, durevole e tollerante ai guasti. Per migliorare la disponibilità, i cluster elastici distribuiscono due nodi per shard collocati in diverse zone di disponibilità. Amazon DocumentDB include diverse funzionalità automatiche che lo rendono una soluzione di database affidabile. Per ulteriori informazioni, consulta Affidabilità di Amazon DocumentDB.

Storage e disponibilità dei cluster elastici

I dati di Amazon DocumentDB sono archiviati in un volume cluster, che è un singolo volume virtuale che utilizza unità a stato solido (SSD). Un volume cluster è composto da sei copie dei tuoi dati, che vengono replicate automaticamente su più zone di disponibilità in una singola regione. AWS Questa replica contribuisce a garantire l'estrema durata dei tuoi dati e a ridurre il rischio di perdita dei dati. Consente inoltre di assicurare che il cluster non sia più disponibile durante un failover perché le copie dei dati sono già presenti in altre zone di disponibilità. Per ulteriori dettagli su storage, alta disponibilità e replica, vedere. Amazon DocumentDB: come funziona

Differenze funzionali tra Amazon DocumentDB 4.0 e i cluster elastici

Esistono le seguenti differenze funzionali tra Amazon DocumentDB 4.0 e i cluster elastici.

- I risultati `top` e `collStats` sono partizionati da frammenti. Per le raccolte frammentate, i dati vengono distribuiti tra più partizioni e `collStats` report aggregati dalle partizioni. `collScans`
- Le statistiche di raccolta da `top` e `collStats` per le raccolte frammentate vengono ripristinate quando il numero di frammenti del cluster viene modificato.
- Il ruolo integrato di backup ora supporta. `serverStatus` Azione: gli sviluppatori e le applicazioni con ruolo di backup possono raccogliere statistiche sullo stato del cluster Amazon DocumentDB.
- Il `SecondaryDelaySecs` campo si sostituisce `slaveDelay` nell'`replSetGetConfig`output.
- Il `hello` comando replaces `isMaster` - `hello` restituisce un documento che descrive il ruolo del cluster elastico.
- L'`$elemMatch`operatore nei cluster elastici corrisponde solo ai documenti nel primo livello di nidificazione di un array. In Amazon DocumentDB 4.0, l'operatore attraversa tutti i livelli prima di restituire i documenti corrispondenti. Ad esempio:

```
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 ] } ] ] }
```

- La proiezione «$» in Amazon DocumentDB 4.0 restituisce tutti i documenti con tutti i campi. Con i cluster elastici, il `find` comando con una proiezione «$» restituisce i documenti che corrispondono al parametro di query contenenti solo il campo che corrisponde alla proiezione «$».
- Nei cluster elastici, i `find` comandi con `$regex` e i parametri di `$options` query restituiscono un errore: «Impossibile impostare le opzioni sia in $regex che in $options».

- Con i cluster elastici, `$indexOfCP` ora restituisce «-1" quando:
  - la sottostringa non si trova in, o `string expression`
  - `start`è un numero maggiore di`end`, o
  - `start`è un numero maggiore della lunghezza in byte della stringa.

  In Amazon DocumentDB 4.0, `$indexOfCP` restituisce «0" quando la `start` posizione è un numero maggiore `end` o la lunghezza in byte della stringa.
- Con i cluster elastici, le operazioni di proiezione`_id fields`, ad esempio`{"_id.nestedField" : 1}`, restituiscono documenti che includono solo il campo proiettato. Nel frattempo, in Amazon DocumentDB 4.0, i comandi di proiezione di campo annidati non filtrano alcun documento.

Inizia a usare i cluster elastici Amazon DocumentDB

Questa sezione introduttiva illustra come creare e interrogare il primo cluster elastico.

Esistono molti modi per connettersi e iniziare a usare Amazon DocumentDB. La seguente procedura è il modo più rapido, semplice e facile per gli utenti di iniziare a utilizzare il nostro potente database di documenti. Questa guida consente AWS CloudShell di connettere e interrogare il cluster Amazon DocumentDB direttamente da. Console di gestione AWS I nuovi clienti idonei per il piano AWS gratuito possono utilizzare Amazon DocumentDB e CloudShell gratuitamente. Se il tuo AWS CloudShell ambiente o il cluster elastico di Amazon DocumentDB utilizza risorse oltre il livello gratuito, ti verranno addebitate le AWS tariffe normali per tali risorse. Questa guida ti aiuterà a iniziare a usare Amazon DocumentDB in meno di 5 minuti.

Argomenti

- Prerequisiti

- Fase 1: Creare un cluster elastico
- Passaggio 2: connettiti al tuo cluster elastico
- Passaggio 3: condividi la tua raccolta, inserisci ed interroga i dati
- Fase 4: Esplora

Prerequisiti

Prima di creare il tuo primo cluster Amazon DocumentDB, devi fare quanto segue:

Crea un account Amazon Web Services (AWS)

Prima di poter iniziare a utilizzare Amazon DocumentDB, devi disporre di un account Amazon Web Services (AWS). L' AWS account è gratuito. Paghi solo per i servizi e le risorse che utilizzi.

Se non ne hai uno Account AWS, completa i seguenti passaggi per crearne uno.

Per iscriverti a Account AWS

1. Aperta https://signin.aws.amazon.com/signup? request_type=registrati.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata o un messaggio di testo e ti verrà chiesto di inserire un codice di verifica attraverso la tastiera del telefono.

   Quando ti iscrivi a un Account AWS, viene creato un. Utente root dell'account AWS L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

Configura le autorizzazioni AWS Identity and Access Management (IAM) necessarie.

L'accesso per gestire le risorse di Amazon DocumentDB come cluster, istanze e gruppi di parametri del cluster richiede credenziali che AWS possono essere utilizzate per autenticare le tue richieste. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per Amazon DocumentDB.

1. Nella barra di ricerca di Console di gestione AWS, inserisci **`IAM`** e seleziona IAM nel menu a discesa.

2. Una volta entrati nella console IAM, seleziona Utenti dal pannello di navigazione.
3. Seleziona il tuo nome utente.
4. Scegli Add Permissions (Aggiungi autorizzazioni).
5. Seleziona Allega direttamente le politiche.
6. `AmazonDocDBElasticFullAccess`Digita la barra di ricerca e selezionala una volta visualizzata nei risultati della ricerca.
7. Scegli Avanti.
8. Scegli Add Permissions (Aggiungi autorizzazioni).

> **Note**
>
> Il tuo AWS account include un VPC predefinito in ogni regione. Se scegli di utilizzare un Amazon VPC, completa i passaggi nell'argomento Creazione di un Amazon VPC nella Amazon VPC User Guide.

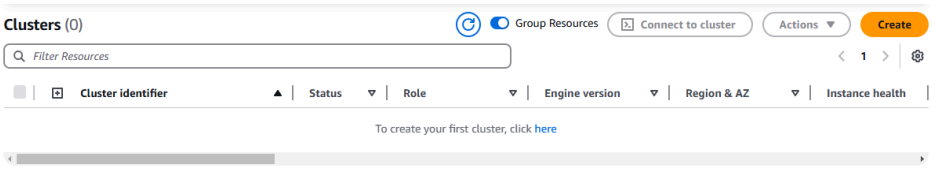
Fase 1: Creare un cluster elastico

In questa sezione spieghiamo come creare un nuovo cluster elastico, utilizzando Console di gestione AWS o AWS CLI con le seguenti istruzioni.

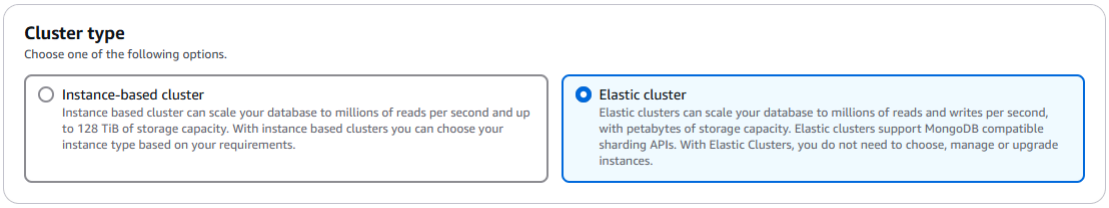
Using the Console di gestione AWS

Per creare una configurazione di cluster elastico utilizzando Console di gestione AWS:

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo https://console.aws.amazon.com/docdb.
2. Nella console di gestione Amazon DocumentDB, in Clusters, scegli Crea.



3. Nella pagina Crea cluster Amazon DocumentDB, nella sezione Tipo di cluster, scegli Elastic cluster.



4. Nella sezione Configurazione, configura quanto segue:

   a. Nel campo Nome del cluster, inserisci un identificatore univoco del cluster (seguendo i requisiti di denominazione sotto il campo).

   b. Nel campo Numero di frammenti, inserisci il numero di frammenti che desideri inserire nel cluster. Il numero massimo di frammenti per cluster è 32.

   > **Note**
   >
   > Verranno distribuiti due nodi per ogni shard. Entrambi i nodi avranno la stessa capacità di shard.

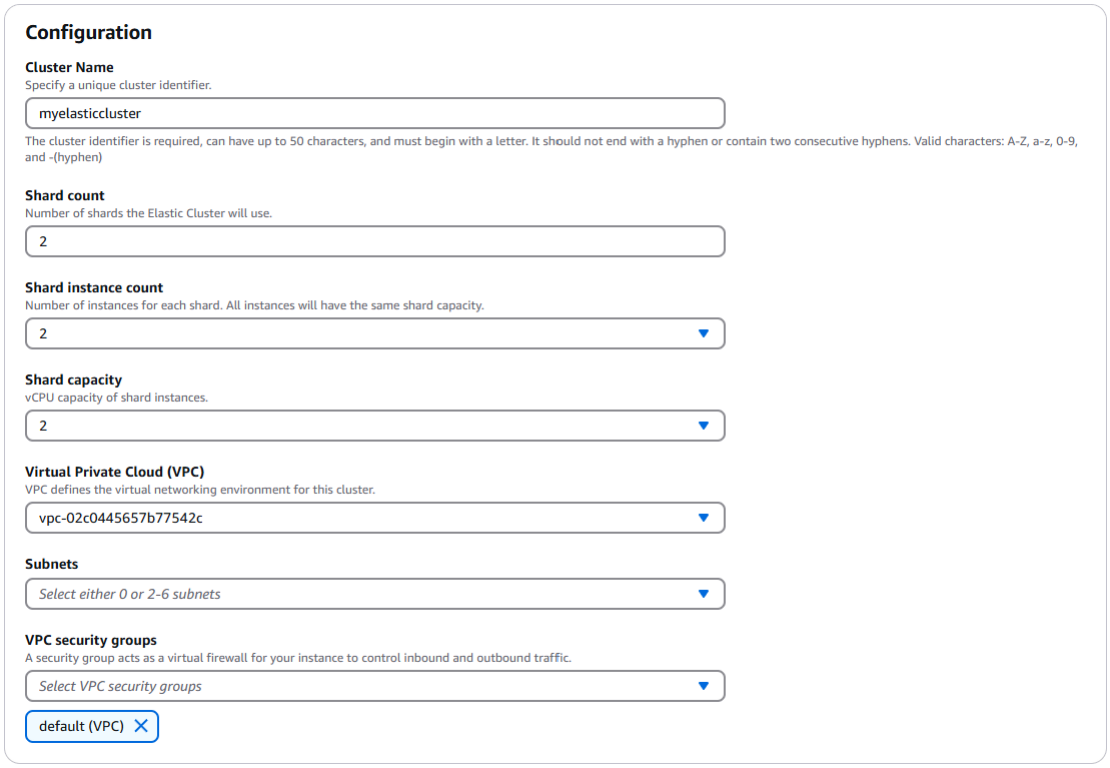
   c. Nel campo Numero di istanze di Shard, scegli il numero di istanze di replica da associare a ogni shard. Il numero massimo di istanze dello shard è 16, con incrementi di 1. Tutte le istanze di replica hanno la stessa capacità di shard definita nel campo seguente. A scopo di test, il valore predefinito di 2 dovrebbe essere sufficiente.

   > **Note**
   >
   > Il numero di istanze di replica si applica a tutti i frammenti del cluster elastico. Un valore di conteggio delle istanze dello shard pari a 1 indica che esiste un'istanza di writer e tutte le istanze aggiuntive sono repliche che possono essere utilizzate per le letture e per migliorare la disponibilità. A scopo di test, il valore predefinito di 2 dovrebbe essere sufficiente.

   d. Nel campo Capacità dello shard, scegli il numero di CPU virtuali (vCPU) da associare a ciascuna istanza dello shard. Il numero massimo di vCPU per istanza di shard è 64. I valori consentiti sono 2, 4, 8, 16, 32, 64. A scopo di test, il valore predefinito di 2 dovrebbe essere sufficiente.

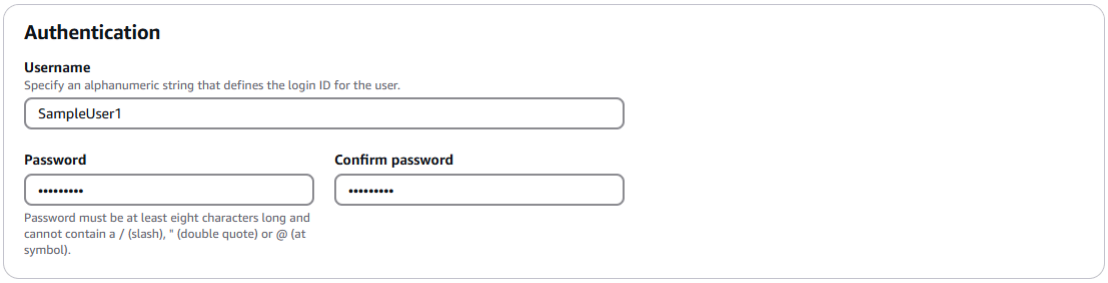
   e. Nel campo Virtual Private Cloud (VPC), scegli un VPC dall'elenco a discesa.

   f. Per le sottoreti e i gruppi di sicurezza VPC, puoi utilizzare le impostazioni predefinite o selezionare tre sottoreti a tua scelta e fino a tre gruppi di sicurezza VPC (minimo uno).



5. Nella sezione Autenticazione, inserisci una stringa che identifichi il nome di accesso dell'utente principale nel campo Nome utente.

   Nel campo Password, inserisci una password univoca conforme alle istruzioni, quindi confermala.



6. Nella sezione Crittografia, mantieni l'impostazione predefinita (Chiave predefinita).

   Facoltativamente, puoi inserire un AWS KMS key ARN che hai creato. Per ulteriori informazioni, consulta Crittografia dei dati inattivi per i cluster elastici Amazon DocumentDB.

> **Important**
>
> La crittografia deve essere abilitata per i cluster elastici.

7. Nella sezione Backup, modifica i campi in base ai requisiti di backup. A scopo di test, è possibile mantenere le impostazioni predefinite.



   a. Periodo di conservazione dei backup: nell'elenco, scegli il numero di giorni per conservare i backup automatici di questo cluster prima di eliminarli.

   b. Finestra di backup: imposta l'ora e la durata giornaliere durante le quali Amazon DocumentDB deve effettuare i backup di questo cluster.

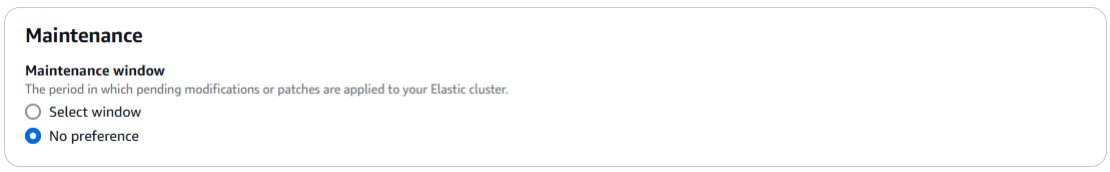
      i. Scegli Seleziona finestra se desideri configurare l'ora e la durata di creazione dei backup.

         Ora di inizio: nel primo elenco, scegli l'ora di inizio (UTC) per l'avvio dei backup automatici. Dal secondo elenco scegli il minuto dell'ora in cui desideri inizino i backup automatici.

         Durata: nell'elenco, scegli il numero di ore da assegnare alla creazione di backup automatici.

      ii. Scegli Nessuna preferenza se desideri che Amazon DocumentDB scelga l'ora e la durata di creazione dei backup.

8. Nella sezione Manutenzione, scegli il giorno, l'ora e la durata in cui le modifiche o le patch vengono applicate al cluster. A scopo di test, puoi mantenere le impostazioni predefinite.



9. Scegli Crea cluster.

Il cluster elastico è ora in fase di provisioning. Il completamento dell'operazione può richiedere alcuni minuti. Puoi connetterti al tuo cluster quando lo stato del cluster elastico appare come Disponibile nell'elenco dei cluster.

Using the AWS CLI

Per creare un cluster elastico utilizzando il AWS CLI, utilizza l'`create-cluster`operazione con i seguenti parametri:

- --cluster-name: obbligatorio. Il nome corrente del cluster di scala elastica immesso durante la creazione o l'ultima modifica.
- --shard-capacity: obbligatorio. Il numero di vCPU assegnate a ogni shard. Il massimo è 64. I valori consentiti sono 2, 4, 8, 16, 32, 64.
- --shard-count: obbligatorio. Il numero di frammenti assegnati al cluster. Il massimo è 32.
- --shard-instance-count—Facoltativo. Il numero di istanze di replica che si applicano a tutti i frammenti di questo cluster. Il massimo è 16.
- --admin-user-name: obbligatorio. Il nome utente associato all'utente amministratore.
- --admin-user-password: obbligatorio. La password associata all'utente amministratore.
- --auth-type: obbligatorio. Il tipo di autenticazione utilizzato per determinare dove recuperare la password utilizzata per accedere al cluster elastico. I tipi validi sono `PLAIN_TEXT` o. `SECRET_ARN`
- --vpc-security-group-ids—Facoltativo. Configura un elenco di gruppi di sicurezza EC2 VPC da associare a questo cluster.
- --preferred-maintenance-window—Facoltativo. Configura l'intervallo di tempo settimanale durante il quale può avvenire la manutenzione del sistema, in Universal Coordinated Time (UTC).

  Il formato è:`ddd:hh24:mi-ddd:hh24:mi`. Giorni validi (ddd): lun, mar, mer, gio, ven, sab, dom

  L'impostazione predefinita è una finestra di 30 minuti selezionata a caso da un intervallo di 8 ore per ciascuna regione Amazon Web Services, che si verifica in un giorno della settimana casuale.

  Finestra minima di 30 minuti.
- --kms-key-id—Opzionale. Configurare l'identificatore chiave KMS per un cluster crittografato.

L'identificatore della chiave KMS è Amazon Resource Name (ARN) per la AWS KMS chiave di crittografia. Se stai creando un cluster utilizzando lo stesso account Amazon Web Services che possiede la chiave di crittografia KMS utilizzata per crittografare il nuovo cluster, puoi utilizzare l'alias della chiave KMS anziché l'ARN per la chiave di crittografia KMS.

Se non è specificata una chiave di crittografia KmsKeyId e se il `StorageEncrypted` parametro è vero, Amazon DocumentDB utilizza la chiave di crittografia predefinita.

- --preferred-backup-window—Facoltativo. L'intervallo di tempo giornaliero preferito durante il quale vengono creati i backup automatici. L'impostazione predefinita è una finestra di 30 minuti selezionata a caso da un intervallo di tempo di 8 ore per ciascuno. Regione AWS
- --backup-retention-period—Facoltativo. Il numero di giorni durante i quali vengono conservati i backup automatici. Il valore predefinito è 1.
- --storage-encrypted—Facoltativo. Configura se il cluster è crittografato o meno.

  `--no-storage-encrypted`specifica che il cluster non è crittografato.
- --subnet-ids—Facoltativo. Configurare gli ID delle sottoreti di rete.

Nel seguente esempio, sostituisci ciascuno *`user input placeholder`* con le tue informazioni.

> **Note**
>
> Gli esempi seguenti includono la creazione di una chiave KMS specifica. Per utilizzare la chiave KMS predefinita, non includete il `--kms-key-id` parametro.

Per Linux, macOS o 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
```

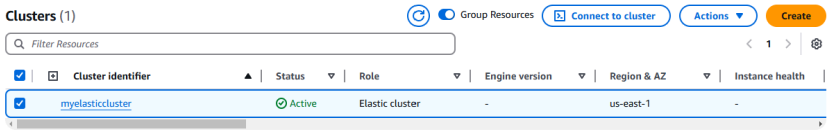
Per Windows:

```
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
```

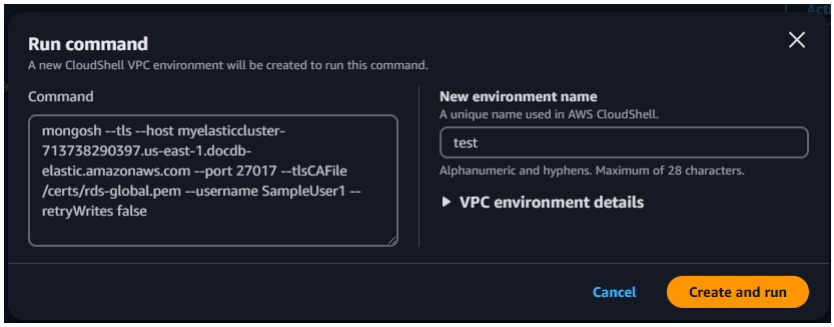
Passaggio 2: connettiti al tuo cluster elastico

Connettiti al tuo cluster elastico Amazon DocumentDB utilizzando AWS CloudShell.

1. Nella console di gestione Amazon DocumentDB, in Clusters, individua il cluster elastico che hai creato. Scegli il tuo cluster facendo clic sulla casella di controllo accanto ad esso.



2. Scegli Connetti al cluster (che si trova accanto al menu a discesa Azioni). Questo pulsante è abilitato solo dopo aver fatto clic sulla casella di controllo accanto al cluster e lo stato del cluster viene visualizzato come Disponibile. Viene visualizzata la schermata del comando CloudShell Esegui.
3. Nel campo Nome nuovo ambiente, inserisci un nome univoco, ad esempio «test» e scegli Crea ed esegui. I dettagli dell'ambiente VPC vengono configurati automaticamente per il database Amazon DocumentDB.



4. Quando richiesto, inserisci la password che hai creato nella fase 1: creazione di un cluster elastico Amazon DocumentDB (sottofase 5).



   Dopo aver inserito la password e ricevuto la richiesta`direct: mongos] <env-name>>`, la connessione al cluster Amazon DocumentDB è stata completata

> **Note**
>
> Per informazioni sulla risoluzione dei problemi, consulta Troubleshooting Amazon DocumentDB.

Passaggio 3: condividi la tua raccolta, inserisci ed interroga i dati

I cluster elastici aggiungono il supporto per lo sharding in Amazon DocumentDB. Ora che sei connesso al tuo cluster, puoi condividere il cluster, inserire dati ed eseguire alcune query.

1. Per condividere una raccolta, inserisci quanto segue:

```
sh.shardCollection("db.Employee1" , { "Employeeid" : "hashed" })
```

2. Per inserire un singolo documento, inserisci quanto segue:

```
db.Employee1.insertOne({"Employeeid":1, "Name":"Joe", "LastName": "Bruin", "level":
 1 })
```

Viene visualizzato il seguente output:

```
WriteResult({ "nInserted" : 1 })
```

3. Per leggere il documento che hai scritto, inserisci il `findOne()` comando (restituisce un singolo documento):

```
db.Employee1.findOne()
```

Viene visualizzato il seguente output:

```
{
"_id" : ObjectId("61f344e0594fe1a1685a8151"),
"EmployeeID" : 1,
"Name" : "Joe",
"LastName" : "Bruin",
"level" : 1
}
```

4. Per eseguire qualche altra domanda, considera un caso d'uso del profilo di gioco. Innanzitutto, inserisci alcune voci in una raccolta intitolata «Employee». Immetti i seguenti dati:

```
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 }
])
```

Viene visualizzato il seguente risultato:

```
{ acknowledged: true,
     insertedIds: {
        '0': ObjectId('679d02cd6b5a0581be78bcbd'),
        '1': ObjectId('679d02cd6b5a0581be78bcbe'),
        '2': ObjectId('679d02cd6b5a0581be78bcbf'),
        '3': ObjectId('679d02cd6b5a0581be78bcc0')
```

```
    }
}
```

5. Per restituire tutti i documenti della raccolta dei profili, immettete il comando `find ()`:

```
db.Employee.find()
```

   Vengono visualizzati i dati inseriti nel passaggio 4.

6. Per interrogare un singolo documento, includi un filtro (ad esempio: «Katie»). Immetti i seguenti dati:

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

   Viene visualizzato il seguente output:

```
[

   {

     _id: ObjectId('679d02cd6b5a0581be78bcc0'),

     Employeeid: 4,

     name: 'Katie',

     lastname: 'Schaper',

     level: 3

   }

]
```

7. Per trovare un profilo e modificarlo, immettete il `findAndModify` comando. In questo esempio, al dipendente «Matt» viene assegnato un livello più alto di «14":

```
db.Employee.findAndModify({
    query: { "Employeeid" : 1, "name" : "Matt"},
    update: { "Employeeid" : 1, "name" : "Matt", "lastname" : "Winkle", "level" :
 14 }
})
```

Viene visualizzato il seguente risultato (si noti che il livello non è ancora cambiato):

```
{

   _id: ObjectId('679d02cd6b5a0581be78bcbd'),

   Employeeid: 1,

   name: 'Matt',

   lastname: 'Winkle',

   level: 12

}
```

8. Per verificare l'aumento del livello, inserisci la seguente query:

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

Viene visualizzato il seguente output:

```
[
   {

   _id: ObjectId('679d02cd6b5a0581be78bcbd'),

   Employeeid: 1,

   name: 'Matt',

   lastname: 'Winkle',

   level: 14

   }
]
```

Fase 4: Esplora

Complimenti! Hai completato con successo la procedura introduttiva per i cluster elastici Amazon DocumentDB.

Qual è il prossimo passo? Scopri come sfruttare appieno questo database con alcune delle sue funzionalità più popolari:

- Best practice per i cluster elastici Amazon DocumentDB
- Gestione dei cluster elastici Amazon DocumentDB

> **Note**
>
> Il cluster elastico che hai creato con questa procedura introduttiva continuerà a generare costi a meno che tu non lo elimini. Per indicazioni, consulta Eliminazione di un cluster elastico.

Best practice per i cluster elastici Amazon DocumentDB

Scopri le migliori pratiche per lavorare con i cluster elastici Amazon DocumentDB. Tutte le best practice per i cluster Amazon DocumentDB basati su istanze si applicano anche ai cluster elastici. Questa sezione viene continuamente aggiornata man mano che vengono identificate nuove best practice.

Argomenti

- Scelta di chiavi condivise
- Gestione delle connessioni
- Raccolte non condivise
- Scalabilità dei cluster elastici
- Monitoraggio dei cluster elastici

Scelta di chiavi condivise

L'elenco seguente descrive le linee guida per la creazione di chiavi shard.

- Usa una chiave hash distribuita in modo uniforme per distribuire i tuoi dati su tutti gli shard del cluster (evita i tasti di scelta rapida).
- Usa la tua chiave shard in tutte le richieste read/update /delete per evitare le query scatter gather.
- Evita le chiavi shard annidate quando esegui operazioni /delete. read/update
- Quando esegui operazioni batch, imposta su false in modo `ordered` che tutti gli shard possano essere eseguiti in parallelo e migliorare le latenze.

Gestione delle connessioni

L'elenco seguente descrive le linee guida per la gestione delle connessioni al database.

- Monitora il numero di connessioni e la frequenza con cui vengono aperte e chiuse nuove connessioni.
- Distribuisci le connessioni su tutte le sottoreti nella configurazione dell'applicazione. Se il cluster è configurato in più sottoreti ma si utilizza solo un sottoinsieme delle sottoreti, è possibile che il numero massimo di connessioni consentite sia limitato.

Raccolte non condivise

Quanto segue descrive una linea guida per le raccolte non condivise.

- Quando lavorate con raccolte non condivise, per distribuire il carico, provate a mantenere le raccolte non frammentate altamente utilizzate su database diversi. I cluster elastici di Amazon DocumentDB posizionano i database su diversi shard e co-localizzano le raccolte non frammentate per lo stesso database sullo stesso shard.

Scalabilità dei cluster elastici

L'elenco seguente descrive le linee guida per il ridimensionamento dei cluster elastici.

- Le operazioni di scalabilità possono causare un breve periodo di errori intermittenti nel database e nella rete. Se possibile, evita il ridimensionamento nelle ore di punta. Provate a scalare durante le finestre di manutenzione.
- È preferibile aumentare o diminuire la capacità degli shard (modificando il numero di vCPU per shard) per aumentare l'elaborazione rispetto all'aumento o alla diminuzione del numero di shard, poiché è più veloce e ha una durata inferiore degli errori intermittenti del database e della rete.

- Quando si prevede una crescita, è preferibile aumentare il numero di shard anziché scalare la capacità degli shard. Ciò consente di scalare il cluster aumentando la capacità degli shard per gli scenari in cui è necessario scalare rapidamente.
- Monitora le politiche relative ai tentativi sul lato client e riprova con backoff e jitter esponenziali per evitare di sovraccaricare il database in caso di errori durante la scalabilità.
- La modifica del numero di shard può richiedere del tempo perché i dati devono essere spostati in modo sicuro da uno shard all'altro. Per ridurre al minimo il rischio di errori, non interrompere o apportare altre modifiche allo stato del cluster mentre la modifica è in corso.

Monitoraggio dei cluster elastici

L'elenco seguente descrive le linee guida per il monitoraggio dei cluster elastici.

- Tieni traccia del rapporto tra picco e media delle tue metriche per shard per determinare se stai generando traffico irregolare (se hai un punto critico). key/hot Le metriche chiave per monitorare i rapporti tra picco e media sono:
  - `PrimaryInstanceCPUUtilization`
    - Questo può essere monitorato a livello di frammento.
    - A livello di cluster è possibile monitorare l'inclinazione media fino a p99.
  - `PrimaryInstanceFreeableMemory`
    - Questo può essere monitorato a livello di frammento.
    - A livello di cluster è possibile monitorare l'inclinazione media fino a p99.
  - `DatabaseCursorsMax`
    - Questo dovrebbe essere monitorato a livello di frammento per determinare l'inclinazione.
  - `Documents-Inserted/Updated/Returned/Deleted`
    - Questo dovrebbe essere monitorato a livello per frammento per determinare l'inclinazione.

Gestione dei cluster elastici Amazon DocumentDB

Per gestire un cluster elastico Amazon DocumentDB, è necessario disporre di una policy IAM con le autorizzazioni appropriate del piano di controllo Amazon DocumentDB. Queste autorizzazioni consentono di creare, modificare ed eliminare i cluster. La DocumentDBFullAccess policy di Amazon fornisce tutte le autorizzazioni necessarie per amministrare un cluster elastico Amazon DocumentDB.

I seguenti argomenti mostrano come eseguire varie attività quando si lavora con i cluster elastici Amazon DocumentDB.

Argomenti

- Modifica delle configurazioni dei cluster elastici
- Monitoraggio di un cluster elastico
- Eliminazione di un cluster elastico
- Gestione delle istantanee dei cluster elastici
- Arresto e avvio di un cluster elastico Amazon DocumentDB
- Manutenzione dei cluster elastici di Amazon DocumentDB

Modifica delle configurazioni dei cluster elastici

In questa sezione spieghiamo come modificare un cluster elastico, utilizzando Console di gestione AWS o AWS CLI con le seguenti istruzioni.

Un uso principale della modifica del cluster consiste nel ridimensionare i frammenti aumentando o diminuendo il numero di shard count and/or , la capacità di calcolo degli shard.

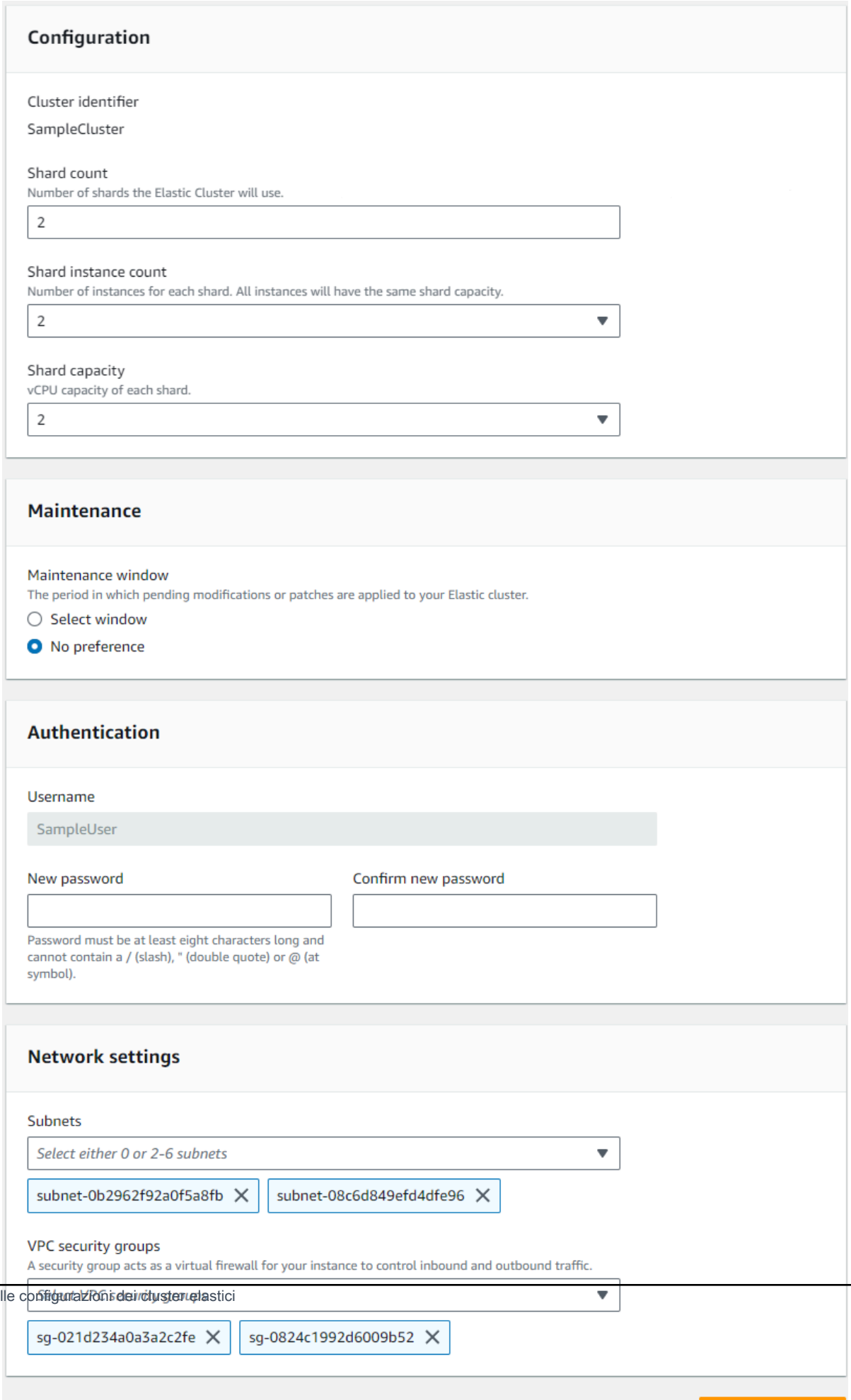
Using the Console di gestione AWS

Per modificare una configurazione elastica del cluster utilizzando: Console di gestione AWS

1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.
2. Nel pannello di navigazione scegliere Cluster.

> **Tip**
>
> Se non vedi il pannello di navigazione sul lato sinistro dello schermo, scegli l'icona del menu nell'angolo in alto a sinistra del pannello di navigazione.

3. Scegli il nome del cluster che desideri modificare nella colonna Identificatore del cluster.
4. Scegli Modifica.
5. Modifica i campi che desideri modificare, quindi seleziona Modifica cluster.


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.
sg-021d234a0a3a2c2fe
sg-0824c1992d6009b52

> **Note**
>
> In alternativa, puoi accedere alla finestra di dialogo Modifica cluster accedendo alla pagina Clusters, selezionando la casella accanto al cluster, scegliendo Azioni, quindi Modifica.

Using the AWS CLI

Per modificare una configurazione di cluster elastico utilizzando il AWS CLI, utilizza l'`update-cluster`operazione con i seguenti parametri:

- **`--cluster-arn`**: obbligatorio. L'identificatore ARN del cluster che si desidera modificare.
- **`--shard-capacity`**—Facoltativo. Il numero di vCPU assegnate a ogni shard. Il massimo è 64. I valori consentiti sono 2, 4, 8, 16, 32, 64.
- **`--shard-count`**—Facoltativo. Il numero di frammenti assegnati al cluster. Il massimo è 32.
- **`--shard-instance`**-Count: opzionale. Il numero di istanze di replica che si applicano a tutti i frammenti di questo cluster. Il massimo è 16.
- **`--auth-type`**—Facoltativo. Il tipo di autenticazione utilizzato per determinare dove recuperare la password utilizzata per accedere al cluster elastico. I tipi validi sono `PLAIN_TEXT` o. `SECRET_ARN`
- **`--admin-user-password`**—Facoltativo. La password associata all'utente amministratore.
- **`--vpc-security-group-ids`**—Facoltativo. Configura un elenco di gruppi di sicurezza Amazon EC2 e Amazon Virtual Private Cloud (VPC) da associare a questo cluster.
- **`--preferred-maintenance-window`**—Facoltativo. Configura l'intervallo di tempo settimanale durante il quale può avvenire la manutenzione del sistema, in Universal Coordinated Time (UTC)

  Il formato è:`ddd:hh24:mi-ddd:hh24:mi`. Giorni validi (ddd): lun, mar, mer, gio, ven, sab, dom

  L'impostazione predefinita è una finestra di 30 minuti selezionata a caso da un intervallo di 8 ore per ciascuna regione Amazon Web Services, che si verifica in un giorno della settimana casuale.

  Finestra minima di 30 minuti.

- **`--subnet-ids`**—Opzionale. Configurare gli ID delle sottoreti di rete.

Nel seguente esempio, sostituisci ciascuno *`user input placeholder`* con le tue informazioni.

Per Linux, macOS o 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
```

Per 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
```

Per monitorare lo stato del cluster elastico dopo la modifica, vedi Monitoraggio di un cluster elastico.

Monitoraggio di un cluster elastico

In questa sezione, spieghiamo come monitorare il cluster elastico, utilizzando Console di gestione AWS o AWS CLI con le seguenti istruzioni.

Using the Console di gestione AWS

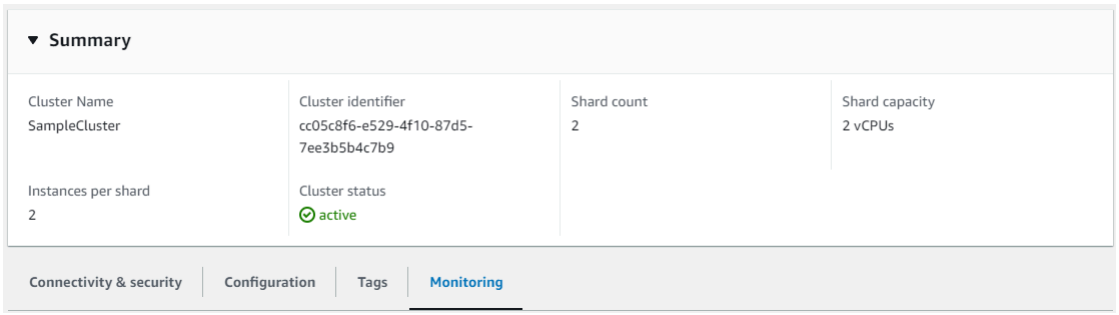
Per monitorare una configurazione di cluster elastico utilizzando Console di gestione AWS:

1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.

2. Nel pannello di navigazione scegliere Cluster.

> **Tip**
>
> Se non vedi il pannello di navigazione sul lato sinistro dello schermo, scegli l'icona del menu nell'angolo in alto a sinistra del pannello di navigazione.

3. Scegli il nome del cluster che desideri monitorare nella colonna Identificatore del cluster.
4. Scegli la scheda Monitoraggio.



CloudWatch Vengono visualizzati alcuni grafici di Amazon per le seguenti categorie di monitoraggio:

- Utilizzo delle risorse
- Throughput
- Operazioni
- Sistema

Puoi anche accedere ad Amazon CloudWatch tramite il Console di gestione AWS per configurare il tuo ambiente di monitoraggio per i tuoi cluster elastici.

Using the AWS CLI

Per monitorare una specifica configurazione di cluster elastici utilizzando il AWS CLI, utilizza l'`get-cluster`operazione con i seguenti parametri:

- **`--cluster-arn`**: obbligatorio. L'identificatore ARN del cluster per il quale desideri informazioni.

Nell'esempio seguente, sostituisci ciascuno *`user input placeholder`* con le tue informazioni.

Per Linux, macOS o Unix:

```
aws docdb-elastic get-cluster \
    --cluster-arn arn:aws:docdb-elastic:us-west-2:123456789012:cluster:/68ffcdf8-e3af-40a3-91e4-24736f2dacc9
```

Per Windows:

```
aws docdb-elastic get-cluster ^
    --cluster-arn arn:aws:docdb:-elastic:us-west-2:123456789012:cluster:/68ffcdf8-e3af-40a3-91e4-24736f2dacc9
```

L'output di questa operazione è simile al seguente:

```
"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",
        ...
 }
```

Per ulteriori informazioni, consulta `DescribeClusterSnapshot` Amazon DocumentDB Resource Management API Reference.

Per visualizzare i dettagli di tutti i cluster elastici che utilizzano il AWS CLI, utilizza l'`list-clusters`operazione con i seguenti parametri:

- **`--next-token`**—Facoltativo. Se l'output del numero di elementi (`--max-results`) è inferiore rispetto al numero totale di elementi restituito dalle chiamate API sottostanti, l'output include `NextToken`, che può essere trasferito a un comando successivo per recuperare il set di elementi successivo.

- **`--max-results`**—Facoltativo. Il numero totale di elementi da restituire nell'output del comando. Se esistono più risultati rispetto al `max-results` valore specificato, nella risposta viene incluso un token di paginazione (`next-token`) in modo che i risultati rimanenti possano essere recuperati.
  - Impostazione predefinita: 100
  - Minimo 20, massimo 100

Nel seguente esempio, sostituisci ciascuno *`user input placeholder`* con le tue informazioni.

Per Linux, macOS o Unix:

```
aws docdb-elastic list-clusters \
    --next-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ== \
    --max-results 2
```

Per Windows:

```
aws docdb-elastic list-clusters ^
    --next-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ== ^
    --max-results 2
```

L'output di questa operazione è simile al seguente:

```
{
   "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"
```

```
        }
    ]
}
```

Eliminazione di un cluster elastico

In questa sezione spieghiamo come eliminare un cluster elastico, utilizzando Console di gestione AWS o AWS CLI con le seguenti istruzioni.

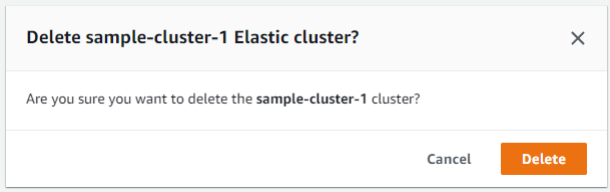
Using the Console di gestione AWS

Per eliminare una configurazione di cluster elastico utilizzando Console di gestione AWS:

1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.
2. Nel pannello di navigazione scegliere Cluster.

> **Tip**
>
> Se non vedi il pannello di navigazione sul lato sinistro dello schermo, scegli l'icona del menu nell'angolo in alto a sinistra del pannello di navigazione.

3. Nella tabella con l'elenco dei cluster, seleziona la casella di controllo a sinistra del nome del cluster che desideri eliminare, quindi scegli Azioni. Dal menu a discesa, scegli Delete (Elimina).
4. Nel cluster elastico Elimina «nome del cluster»? finestra di dialogo, scegli Elimina.



Sono necessari alcuni minuti per l'eliminazione del cluster. Per monitorare lo stato del cluster, consulta Monitoraggio dello stato di un cluster Amazon DocumentDB.

Using the AWS CLI

Per eliminare un cluster elastico utilizzando il AWS CLI, utilizzare l'`delete-cluster`operazione con i seguenti parametri:

- **`--cluster-arn`**: obbligatorio. L'identificatore ARN del cluster che desideri eliminare.
- **`--no-skip-final-backup`**—Facoltativo. Se si desidera un backup finale, è necessario includere questo parametro con un nome per il backup finale. È necessario includere `--final-backup-identifier` o `--skip-final-backup`.
- **`--skip-final-backup`**—Facoltativo. Utilizza questo parametro solo se non desideri eseguire un backup finale prima di eliminare il cluster. L'impostazione predefinita prevede l'acquisizione di uno snapshot finale

I seguenti esempi di AWS CLI codice eliminano un cluster con un ARN di arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster con un backup finale.

Nel seguente esempio, sostituisci ciascuno *`user input placeholder`* con le tue informazioni..

Per Linux, macOS o 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
```

Per 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
```

I seguenti esempi di AWS CLI codice eliminano un cluster con un ARN di arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster senza eseguire un backup finale.

Nel seguente esempio, sostituisci ciascuno *`user input placeholder`* con le tue informazioni.

Per Linux, macOS o Unix:

```
aws docdb-elastic delete-cluster \
```

```
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster \
    --skip-final-backup \
```

Per Windows:

```
aws docdb-elastic delete-cluster ^
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster ^
    --skip-final-backup ^
```

L'output dell'`delete-cluster`operazione è una visualizzazione del cluster che stai eliminando.

Sono necessari alcuni minuti per l'eliminazione del cluster. Per monitorare lo stato del cluster, consulta Monitoraggio dello stato di un cluster Amazon DocumentDB.

Gestione delle istantanee dei cluster elastici

Le istantanee manuali possono essere scattate dopo la creazione di un cluster elastico. I backup automatici vengono creati nel momento in cui viene creata l'istantanea del cluster elastico.

> **Note**
>
> Il cluster elastico deve essere nello `Available` stato in cui è possibile scattare un'istantanea manuale.

Questa sezione spiega come creare, visualizzare, ripristinare ed eliminare le istantanee del cluster elastico.

I seguenti argomenti mostrano come eseguire varie attività quando si lavora con gli snapshot dei cluster elastici Amazon DocumentDB.

Argomenti

- Creazione manuale di un'istantanea del cluster elastico
- Visualizzazione di un'istantanea di un cluster elastico
- Ripristino di un cluster elastico da un'istantanea

- Copiare un'istantanea di un cluster elastico
- Eliminazione di un'istantanea di un cluster elastico
- Gestione di un backup automatico dello snapshot di un cluster elastico

Creazione manuale di un'istantanea del cluster elastico

In questa sezione spieghiamo come creare un'istantanea manuale del cluster elastico, utilizzando Console di gestione AWS o AWS CLI con le seguenti istruzioni.

Using the Console di gestione AWS

Per creare un'istantanea manuale del cluster elastico utilizzando: Console di gestione AWS

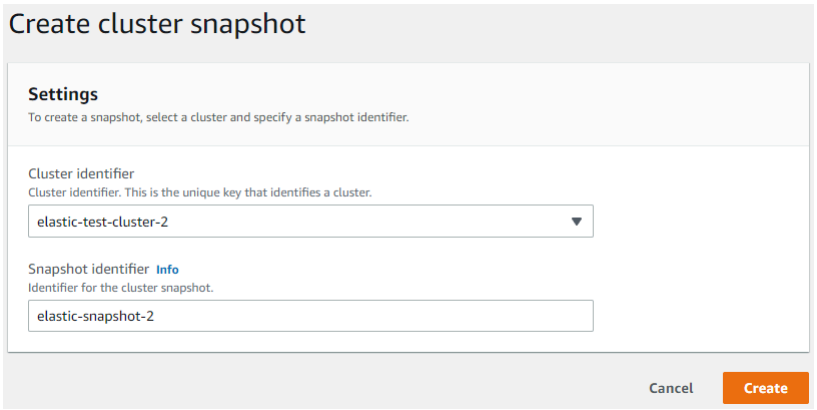
1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.
2. Nel pannello di navigazione, selezionare Snapshots (Snapshot).

> **Tip**
>
> Se non vedi il pannello di navigazione sul lato sinistro dello schermo, scegli l'icona del menu nell'angolo in alto a sinistra del pannello di navigazione.

3. Nella pagina Snapshots (Snapshot) scegli Create (Crea).
4. Nella pagina Crea un'istantanea del cluster, nel campo Identificatore del cluster, scegli il tuo cluster elastico dall'elenco a discesa.

   Nel campo Identificatore istantanea, inserisci un identificatore univoco per il tuo cluster elastico.

   Scegli Create (Crea).



> **Note**
>
> In alternativa, puoi accedere alla finestra di dialogo Crea istantanea del cluster andando alla pagina Clusters, selezionando la casella accanto al cluster, quindi scegliendo Azioni, quindi Scatta istantanea.

Lo snapshot del tuo cluster elastico è ora in fase di provisioning. Il completamento dell'operazione può richiedere alcuni minuti. È possibile visualizzare e ripristinare un'istantanea quando lo stato è indicato `Available` nell'elenco Istantanee.

Using the AWS CLI

Per creare un'istantanea manuale del cluster elastico utilizzando il AWS CLI, utilizza l'`create-cluster-snapshot`operazione con i seguenti parametri:

- **`--snapshot-name`**: obbligatorio. Il nome dello snapshot del cluster che desideri creare.
- **`--cluster-arn`**: obbligatorio. L'identificatore ARN del cluster di cui si desidera creare un'istantanea.

Nel seguente esempio, sostituisci ciascuno *`user input placeholder`* con le tue informazioni.

Per Linux, macOS o 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
```

Per 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
```

Visualizzazione di un'istantanea di un cluster elastico

In questa sezione spieghiamo come visualizzare le informazioni sullo snapshot del cluster elastico, utilizzando Console di gestione AWS o AWS CLI con le seguenti istruzioni.

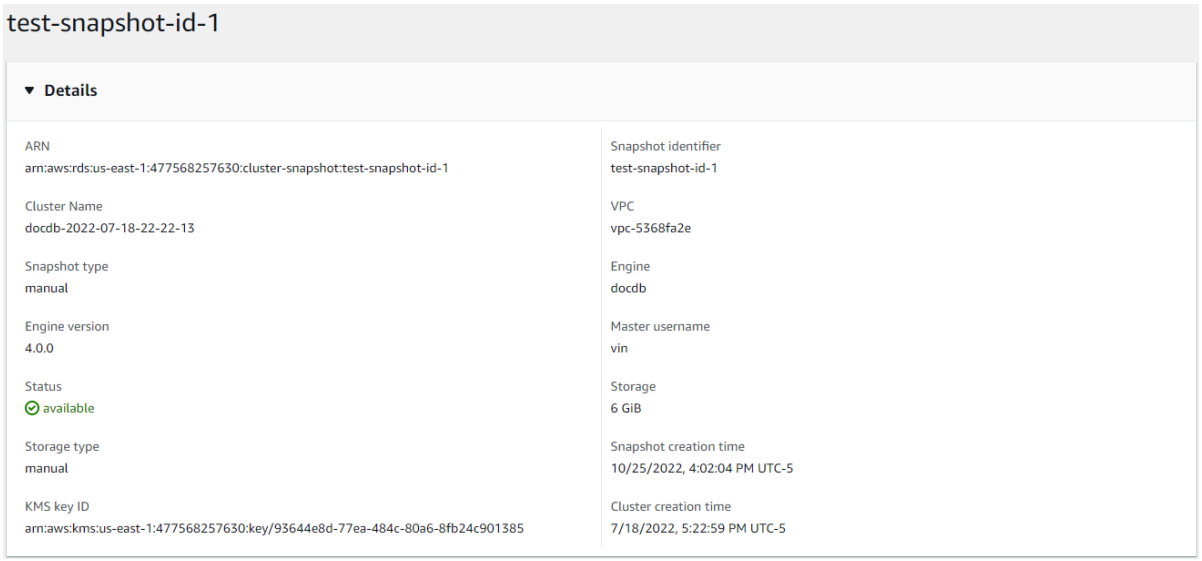
Using the Console di gestione AWS

Per visualizzare le informazioni su uno specifico snapshot di un cluster elastico utilizzando: Console di gestione AWS

1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.
2. Nel pannello di navigazione, selezionare Snapshots (Snapshot).

> **Tip**
>
> Se non vedi il riquadro di navigazione sul lato sinistro dello schermo, scegli l'icona del menu nell'angolo in alto a sinistra del pannello di navigazione.

3. Nella pagina Istantanee, scegli la tua istantanea dall'elenco facendo clic sul nome nella colonna dell'identificatore dell'istantanea.
4. Visualizza le informazioni sulla tua istantanea in Dettagli.



Using the AWS CLI

Per visualizzare le informazioni su uno specifico snapshot di un cluster elastico utilizzando il AWS CLI, utilizza l'`get-cluster-snapshot`operazione con i seguenti parametri:

- **`--snapshot-arn`**: obbligatorio. L'identificatore ARN dello snapshot per il quale si desiderano informazioni.

Nel seguente esempio, sostituisci ciascuno *`user input placeholder`* con le tue informazioni.

Per Linux, macOS o Unix:

```
aws docdb-elastic get-cluster-snapshot \
    --snapshot-arn sampleResourceName
```

Per Windows:

```
aws docdb-elastic get-cluster-snapshot ^
    --snapshot-arn sampleResourceName
```

Per visualizzare le informazioni su uno specifico snapshot di un cluster elastico utilizzando il AWS CLI, utilizza l'`get-cluster-snapshot`operazione con i seguenti parametri:

- **`--snapshot-arn`**: obbligatorio. L'identificatore ARN dello snapshot per il quale si desiderano informazioni.

Nel seguente esempio, sostituisci ciascuno *`user input placeholder`* con le tue informazioni.

Per Linux, macOS o Unix:

```
aws docdb-elastic get-cluster-snapshot \
    --snapshot-arn sampleResourceName
```

Per Windows:

```
aws docdb-elastic get-cluster-snapshot ^
    --snapshot-arn sampleResourceName
```

Per visualizzare le informazioni su tutte le istantanee dei cluster elastici utilizzando il AWS CLI, utilizza l'`list-cluster-snapshots`operazione con i seguenti parametri:

- **`--snapshot-type`**—Facoltativo. Il tipo di istantanee del cluster da restituire. È possibile specificare uno dei seguenti valori:
  - `automated`- Restituisci tutte le istantanee del cluster che Amazon DocumentDB ha creato automaticamente per il tuo account. AWS
  - `manual`- Restituisce tutte le istantanee del cluster che hai creato manualmente per il tuo account. AWS
  - `shared`- Restituisci tutte le istantanee manuali del cluster che sono state condivise con il tuo AWS account.
  - `public`- Restituisce tutte le istantanee del cluster che sono state contrassegnate come pubbliche.
- **`--next-token`**—Facoltativo. Token di paginazione opzionale fornito da una richiesta precedente. Se viene specificato questo parametro, la risposta include solo i record oltre a questo token, fino al valore specificato da`max-results`.
- **`--max-results`**—Facoltativo. Il numero massimo di risultati da includere nella risposta. Se esistono più risultati del `max-results` valore specificato, nella risposta viene incluso un token di paginazione (`next-token`) in modo che i risultati rimanenti possano essere recuperati.
  - Impostazione predefinita: 100
  - Minimo 20, massimo 100

Nel seguente esempio, sostituisci ciascuno *`user input placeholder`* con le tue informazioni.

Per Linux, macOS o Unix:

```
aws docdb-elastic list-cluster-snapshots \
    --snapshot-type  value \
    --next-token  value \
    --max-results 50
```

Per Windows:

```
aws docdb-elastic list-cluster-snapshots ^
    --snapshot-type  value ^
    --next-token  value ^
    --max-results 50
```

Ripristino di un cluster elastico da un'istantanea

In questa sezione spieghiamo come ripristinare un cluster elastico da un'istantanea, utilizzando Console di gestione AWS o AWS CLI con le seguenti istruzioni.

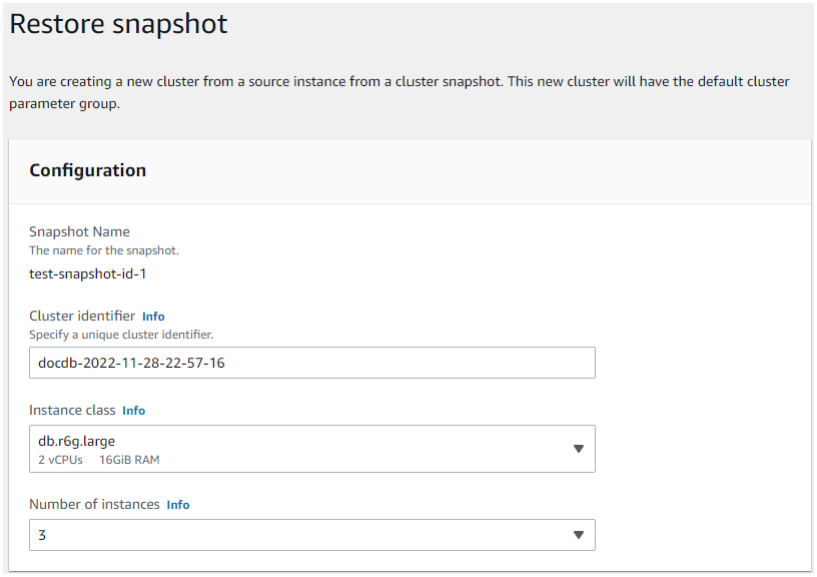
Using the Console di gestione AWS

Per ripristinare un cluster elastico da un'istantanea utilizzando: Console di gestione AWS

1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.
2. Nel pannello di navigazione, selezionare Snapshots (Snapshot).

> **Tip**
>
> Se non vedi il riquadro di navigazione sul lato sinistro dello schermo, scegli l'icona del menu nell'angolo in alto a sinistra del pannello di navigazione.

3. Scegli il pulsante a sinistra dell'istantanea, che desideri utilizzare per ripristinare un cluster, nella colonna Identificatore dello snapshot.
4. Seleziona Operazioni, quindi Ripristina.



5. Nella pagina Restore snapshot, inserisci un nome per il nuovo cluster nel campo Cluster identifier.

> **Note**
>
> Per qualsiasi ripristino manuale delle istantanee, è necessario creare un nuovo cluster.

6. Nel campo Virtual Private Cloud (VPC), scegli un VPC dall'elenco a discesa.
7. Per le sottoreti e i gruppi di sicurezza VPC, puoi utilizzare le impostazioni predefinite o selezionare tre sottoreti a tua scelta e fino a tre gruppi di sicurezza VPC (minimo uno).
8. Se si è soddisfatti della configurazione del cluster, scegliere Restore cluster (Ripristina cluster) e attendere il ripristino del cluster.

Using the AWS CLI

Per ripristinare un cluster elastico da un'istantanea utilizzando il, utilizza l'operazione con i seguenti parametri: AWS CLI`restore-cluster-from-snapshot`

- **`--cluster-name`**: obbligatorio. Il nome corrente del cluster elastico inserito durante la creazione o l'ultima modifica.
- **`--snapshot-arn`**: obbligatorio. L'identificatore ARN dello snapshot utilizzato per ripristinare il cluster.
- **`--vpc-security-group-ids`**—Facoltativo. Uno o più gruppi di sicurezza Amazon EC2 e Amazon Virtual Private Cloud (VPC) da associare al cluster.
- **`--kms-key-id`**—Opzionale. Configurare l'identificatore della chiave KMS per un cluster crittografato.

  L'identificatore della chiave KMS è Amazon Resource Name (ARN) per la chiave di crittografia. AWS KMS Se stai creando un cluster utilizzando lo stesso account Amazon Web Services che possiede la chiave di crittografia KMS utilizzata per crittografare il nuovo cluster, puoi utilizzare l'alias della chiave KMS anziché l'ARN per la chiave di crittografia KMS.

  Se non è specificata una chiave di crittografia KmsKeyId e se il `StorageEncrypted` parametro è vero, Amazon DocumentDB utilizza la chiave di crittografia predefinita.
- **`--subnet-ids`**—Facoltativo. ID della sottorete di rete.

Nel seguente esempio, sostituisci ciascuno *`user input placeholder`* con le tue informazioni..

Per Linux, macOS o 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
```

Per 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
```

Copiare un'istantanea di un cluster elastico

In Amazon DocumentDB, puoi copiare istantanee manuali e automatiche dei cluster elastici all'interno della stessa regione e all'interno dello stesso account. In questa sezione spieghiamo come copiare uno snapshot di un cluster elastico, utilizzando o. Console di gestione AWS AWS CLI

Using the Console di gestione AWS

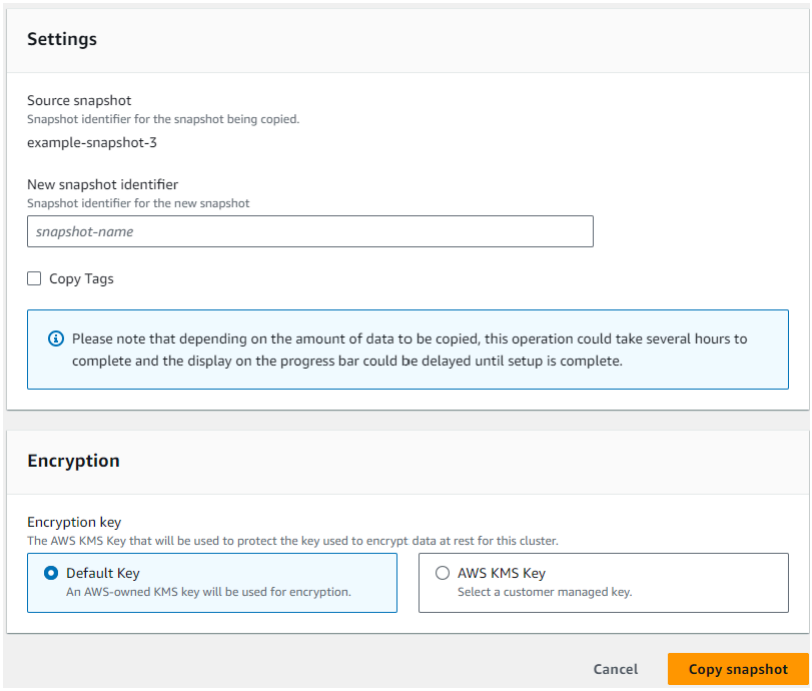
Per copiare un'istantanea di un cluster elastico utilizzando: Console di gestione AWS

1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.
2. Nel pannello di navigazione, selezionare Snapshots (Snapshot).

   > **Tip**
   >
   > Se non vedi il riquadro di navigazione sul lato sinistro dello schermo, scegli l'icona del menu nell'angolo in alto a sinistra del pannello di navigazione.

3. Scegli il pulsante a sinistra dell'istantanea che desideri copiare nella colonna Identificatore dell'istantanea.

4. Scegli Azioni, quindi Copia.



5. Per New snapshot identifier, inserisci il nome del nuovo snapshot.
6. Per Copia tag, seleziona la casella se desideri copiare tutti i tag dallo snapshot del cluster elastico di origine allo snapshot del cluster elastico di destinazione.
7. Per la crittografia, scegli una chiave AWS KMS predefinita o una chiave KMS a tua scelta. La seconda opzione ti consente di selezionare una chiave KMS esistente che hai già creato o di crearne una nuova.
8. Al termine, scegli Copia istantanea.

Using the AWS CLI

Per copiare un'istantanea di un cluster elastico utilizzando il AWS CLI, utilizzate l'`copy-cluster-snapshot`operazione con i seguenti parametri:

- **`--source-db-cluster-snapshot-identifier`**: obbligatorio. L'identificatore dello snapshot del cluster elastico esistente da copiare. Lo snapshot del cluster elastico deve esistere ed essere nello stato disponibile. Questo parametro non distingue tra maiuscole e minuscole.
- **`--target-db-cluster-snapshot-identifier`**: obbligatorio. L'identificatore del nuovo snapshot del cluster elastico da creare a partire dallo snapshot del cluster esistente. Questo parametro non distingue tra maiuscole e minuscole.

Vincoli relativi al nome dello snapshot di destinazione:

- Non può essere il nome di uno snapshot esistente.
- La lunghezza è di [1—63] lettere, numeri o trattini.
- Il primo carattere deve essere una lettera.
- Non può terminare con un trattino o contenere due trattini consecutivi.

Nel seguente esempio, sostituisci ciascuno *user input placeholder* con le tue informazioni.

Per Linux, macOS o Unix:

```
aws docdb-elastic copy-cluster-snapshot \
     --source-cluster-snapshot-arn <sample ARN> \
     --target-cluster-snapshot-name my-target-copied-snapshot
```

Per Windows:

```
aws docdb-elastic copy-cluster-snapshot ^
     --source-cluster-snapshot-arn <sample ARN> ^
     --target-cluster-snapshot-name my-target-copied-snapshot
```

Eliminazione di un'istantanea di un cluster elastico

In questa sezione spieghiamo come eliminare un'istantanea di un cluster elastico, utilizzando o. Console di gestione AWS AWS CLI

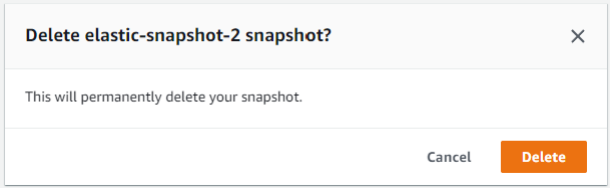
Using the Console di gestione AWS

Per ripristinare un cluster elastico da un'istantanea utilizzando: Console di gestione AWS

1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.
2. Nel pannello di navigazione, selezionare Snapshots (Snapshot).

> **Tip**
>
> Se non vedi il riquadro di navigazione sul lato sinistro dello schermo, scegli l'icona del menu nell'angolo in alto a sinistra del pannello di navigazione.

3. Scegli il pulsante a sinistra dell'istantanea, che desideri utilizzare per ripristinare un cluster, nella colonna Identificatore dello snapshot.
4. Scegli Operazioni, quindi Elimina.



5. Nella finestra di dialogo Elimina istantanea «snapshot-name», scegli Elimina.

Using the AWS CLI

Per eliminare un'istantanea di un cluster elastico utilizzando il AWS CLI, utilizza l'operazione con i seguenti parametri: `delete-cluster-snapshot`

- **`--snapshot-arn`**: obbligatorio. L'identificatore ARN dello snapshot utilizzato per ripristinare il cluster.

Nell'esempio seguente, sostituisci ciascuno *`user input placeholder`* con le tue informazioni..

Per Linux, macOS o Unix:

```
aws docdb-elastic delete-cluster-snapshot \
     --snapshot-arn sampleResourceName
```

Per Windows:

```
aws docdb-elastic delete-cluster-snapshot ^
     --snapshot-arn sampleResourceName
```

Gestione di un backup automatico dello snapshot di un cluster elastico

Amazon DocumentDB scatta istantanee giornaliere dei tuoi cluster elastici. Puoi specificare la finestra di backup preferita e il periodo di conservazione del backup in una configurazione snapshot del cluster elastico nuova o esistente. In questa sezione spieghiamo come impostare i parametri di backup automatici in un'istantanea di un cluster elastico, utilizzando l' Console di gestione AWS opzione or. AWS CLI

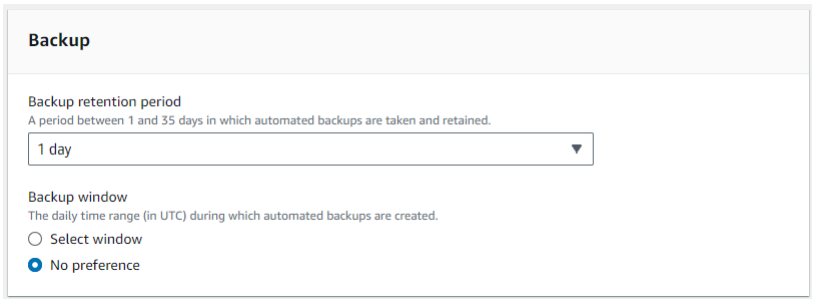
Using the Console di gestione AWS

Per impostare un backup automatico per una nuova istantanea del cluster elastico utilizzando: Console di gestione AWS

1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.
2. Nel pannello di navigazione scegliere Cluster.

> **Tip**
>
> Se non vedi il riquadro di navigazione sul lato sinistro dello schermo, scegli l'icona del menu nell'angolo superiore sinistro del riquadro di navigazione.

3. Scegli il pulsante a sinistra del cluster, per il quale desideri modificare le impostazioni di backup, nella colonna Identificatore del cluster.
4. Scegli Azioni, quindi Modifica.
5. Nella sezione Backup, modifica i campi in base ai requisiti di backup.



   a. Periodo di conservazione dei backup: nell'elenco, scegli il numero di giorni per conservare i backup automatici di questo cluster prima di eliminarli.
   b. Finestra di backup: imposta l'ora e la durata giornaliere durante le quali Amazon DocumentDB deve effettuare i backup di questo cluster.
      i. Scegli Seleziona finestra se desideri configurare l'ora e la durata di creazione dei backup.

         Ora di inizio: nel primo elenco, scegli l'ora di inizio (UTC) per l'avvio dei backup automatici. Dal secondo elenco scegli il minuto dell'ora in cui desideri inizino i backup automatici.

         Durata: nell'elenco, scegli il numero di ore da assegnare alla creazione di backup automatici.

ii. Scegli Nessuna preferenza se desideri che Amazon DocumentDB scelga l'ora e la durata di creazione dei backup.

6. Una volta completato, scegli Modifica cluster.

Using the AWS CLI

Per impostare un backup automatico per una nuova istantanea del cluster elastico utilizzando il AWS CLI, utilizza l'`create-cluster-snapshot`operazione con i seguenti parametri:

- **`--preferred-backup-window`**—Facoltativo. L'intervallo di tempo giornaliero preferito durante il quale vengono creati i backup automatici. L'impostazione predefinita è una finestra di 30 minuti selezionata a caso da un intervallo di tempo di 8 ore per ciascuno. Regione AWS

  Vincoli:

  - Il valore deve essere nel formato `hh24:mi-hh24:mi`.
  - Il valore deve essere nel fuso orario UTC (Universal Coordinated Time).
  - Il valore non deve essere in conflitto con la finestra di manutenzione preferita.
  - Il valore deve essere almeno di 30 minuti.
- **`--backup-retention-period`**—Facoltativo. Il numero di giorni durante i quali vengono conservati i backup automatici. Il valore predefinito è 1.

  Vincoli:

  - È necessario specificare un valore minimo di 1.
  - L'intervallo è compreso tra 1 e 35.

> **Note**
>
> I backup automatici vengono eseguiti solo quando il cluster è in uno stato «attivo».

> **Note**
>
> È inoltre possibile modificare i `backup-retention-period` parametri `preferred-backup-window` e di un cluster elastico esistente utilizzando il `aws docdb-elastic update-cluster` comando.

Nell'esempio seguente, sostituisci ciascuno *user input placeholder* con le tue informazioni.

L'`create-cluster`esempio seguente crea il cluster elastico Amazon DocumentDB *sample-cluster* con un periodo di conservazione per i backup automatici di *7* giorni e una finestra di backup preferita di. *18:00-18:30 UTC*

Per Linux, macOS o 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
```

Per 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
```

Arresto e avvio di un cluster elastico Amazon DocumentDB

L'arresto e l'avvio dei cluster elastici Amazon DocumentDB possono aiutarti a gestire i costi per gli ambienti di sviluppo e test. Invece di creare ed eliminare cluster elastici ogni volta che usi Amazon DocumentDB, puoi interrompere temporaneamente il cluster quando non è necessario. Potrai quindi riavviarlo quando riprenderai i test.

Argomenti

- Panoramica sull'arresto e l'avvio di un cluster elastico
- Operazioni che è possibile eseguire su un cluster elastico interrotto

Panoramica sull'arresto e l'avvio di un cluster elastico

Nei periodi in cui non è necessario un cluster elastico Amazon DocumentDB, puoi arrestare il cluster. È possibile avviare nuovamente il cluster ogni volta che è necessario utilizzarlo. L'avvio e l'arresto semplificano i processi di configurazione e smontaggio dei cluster elastici utilizzati per lo sviluppo, il test o attività simili che non richiedono una disponibilità continua. È possibile arrestare e avviare un cluster elastico utilizzando Console di gestione AWS o AWS CLI con una sola azione.

Mentre il cluster elastico è fermo, il volume di archiviazione del cluster rimane invariato. Vengono addebitati solo i costi per lo storage, gli snapshot manuali e lo storage di backup automatici all'interno della finestra di retention specificata. Amazon DocumentDB avvia automaticamente il cluster elastico dopo sette giorni in modo che non subisca gli aggiornamenti di manutenzione richiesti. Quando il cluster viene avviato dopo sette giorni, inizieranno nuovamente a essere addebitati i costi per l'utilizzo del cluster elastico. Quando il cluster è fermo, non puoi interrogare il volume di archiviazione perché l'interrogazione richiede che il cluster sia nello stato disponibile.

Quando un cluster elastico Amazon DocumentDB viene interrotto, il cluster non può essere modificato in alcun modo. Ciò include l'eliminazione del cluster.

Using the Console di gestione AWS

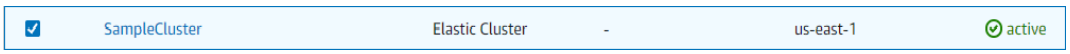
La procedura seguente mostra come interrompere un cluster elastico nello stato disponibile o avviare un cluster elastico interrotto.

Per arrestare o avviare un cluster elastico Amazon DocumentDB

1. Accedi a e apri Console di gestione AWS la console Amazon DocumentDB all'indirizzo. https://console.aws.amazon.com/docdb
2. Nel pannello di navigazione scegliere Cluster.

   > **Tip**
   >
   > Se il riquadro di navigazione non viene visualizzato sul lato sinistro della schermata, scegliere l'icona del menu (≡) nell'angolo in alto a sinistra della pagina.

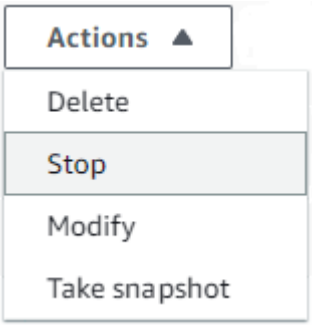
3. Quindi, nell'elenco dei cluster, scegliere il pulsante a sinistra del nome del cluster che si desidera arrestare o avviare.



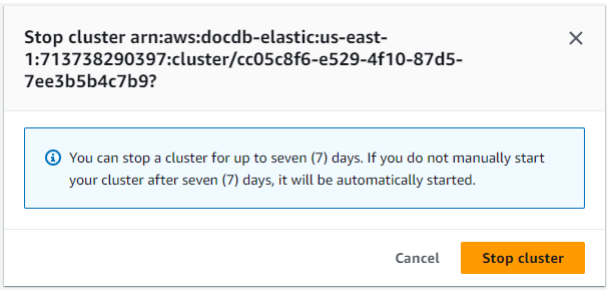
4. Scegliere Actions (Operazioni) e quindi scegliere l'operazione che si desidera eseguire sul cluster.

   - Se si desidera arrestare il cluster e il cluster è disponibile:

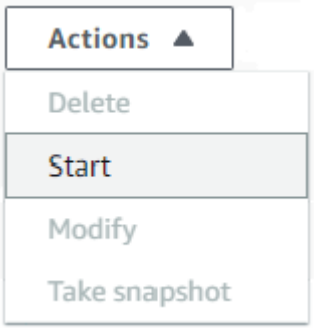
     a. Scegli Stop (Arresta).



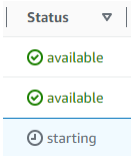
     b. Nella finestra di dialogo di conferma, conferma di voler interrompere il cluster elastico scegliendo Stop cluster oppure, per mantenere il cluster in esecuzione, scegli Annulla.



   - Se si desidera avviare il cluster e il cluster è arrestato, scegliere Start (Avvia).



5. Monitora lo stato del cluster elastico. Se hai avviato il cluster, puoi riprendere a utilizzarlo quando il cluster è disponibile. Per ulteriori informazioni, consulta Determinazione dello stato di un cluster.



Using the AWS CLI

I seguenti esempi di codice mostrano come interrompere un cluster elastico nello stato attivo o disponibile o avviare un cluster elastico interrotto.

Per interrompere un cluster elastico utilizzando il AWS CLI, utilizzare l'`stop-cluster`operazione. Per avviare un cluster arrestato, utilizzare l'operazione `start-cluster`. Entrambe le operazioni consentono di utilizzare il parametro `--cluster-arn`.

Parametro:

- **`--cluster-arn`**: obbligatorio. L'identificatore ARN del cluster elastico che desideri arrestare o avviare.

Example— Per interrompere un cluster elastico utilizzando AWS CLI

Nel seguente esempio, sostituisci ciascuno *`user input placeholder`* con le tue informazioni.

Il codice seguente interrompe il cluster elastico con un ARN pari `arn:aws:docdb-elastic:us-east-1:477568257630:cluster/b9f1d489-6c3e-4764-bb42-da62ceb7bda2` a.

> **Note**
>
> Il cluster elastico deve essere attivo o disponibile.

Per Linux, macOS o Unix:

```
aws docdb-elastic stop-cluster \
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Per Windows:

```
aws docdb-elastic stop-cluster ^
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Example— Per avviare un cluster elastico utilizzando AWS CLI

Nel seguente esempio, sostituisci ciascuno *user input placeholder* con le tue informazioni.

Il codice seguente avvia il cluster elastico con un ARN pari a. `arn:aws:docdb-elastic:us-east-1:477568257630:cluster/b9f1d489-6c3e-4764-bb42-da62ceb7bda2`

> **Note**
>
> Il cluster elastico deve essere attualmente interrotto.

Per Linux, macOS o Unix:

```
aws docdb-elastic start-cluster \
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Per Windows:

```
aws docdb-elastic start-cluster ^
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Operazioni che è possibile eseguire su un cluster elastico interrotto

Non è possibile modificare la configurazione di un cluster elastico Amazon DocumentDB mentre il cluster è fermo. Devi avviare il cluster prima di eseguire questo tipo di operazioni amministrative.

Amazon DocumentDB applica qualsiasi manutenzione programmata al cluster elastico interrotto solo dopo il riavvio. Dopo sette giorni, Amazon DocumentDB avvia automaticamente un cluster elastico interrotto in modo che non rimanga troppo indietro nel suo stato di manutenzione. Al riavvio del cluster elastico, ricominceranno a essere addebitati i frammenti contenuti nel cluster.

Mentre un cluster elastico viene interrotto, Amazon DocumentDB non esegue alcun backup automatico né estende il periodo di conservazione del backup.

Manutenzione dei cluster elastici di Amazon DocumentDB

Argomenti

- Visualizzazione delle azioni di manutenzione del cluster elastico in sospeso
- Aggiornamenti del motore Elastic Cluster
- Aggiornamenti del sistema operativo Elastic Cluster

Periodicamente, Amazon DocumentDB esegue la manutenzione sulle risorse dei cluster elastici di Amazon DocumentDB. La manutenzione comporta spesso aggiornamenti al motore del database (manutenzione elastica del cluster) o al sistema operativo sottostante del cluster elastico (aggiornamenti del sistema operativo). Gli aggiornamenti del motore di database sono patch obbligatorie e includono correzioni di sicurezza, correzioni di bug e miglioramenti al motore di database. Sebbene la maggior parte delle patch del sistema operativo siano opzionali, se non le applichi per un po' di tempo, la patch potrebbe essere obbligatoria e applicata automaticamente per mantenere il livello di sicurezza. Applica gli aggiornamenti del sistema operativo ai tuoi cluster elastici Amazon DocumentDB non appena sono disponibili.

Le patch del motore di database richiedono la disattivazione dei cluster elastici Amazon DocumentDB per un breve periodo. Una volta disponibili, queste patch vengono automaticamente programmate per essere applicate durante una prossima finestra di manutenzione programmata del cluster elastico Amazon DocumentDB.

I cluster elastici hanno le rispettive finestre di manutenzione. Le modifiche ai cluster elastici che hai scelto di non applicare immediatamente vengono applicate durante la finestra di manutenzione. Per impostazione predefinita, quando si crea un cluster elastico, Amazon DocumentDB assegna una finestra di manutenzione per il cluster elastico. Puoi scegliere la finestra di manutenzione quando crei un cluster elastico. che può anche essere modificata in qualsiasi momento per adeguarsi alla pianificazione o alle prassi del business. In genere è consigliabile scegliere finestre di manutenzione con impatto minimo sull'applicazione (per esempio, in serata o nel week end).

Visualizzazione delle azioni di manutenzione del cluster elastico in sospeso

Puoi verificare se è disponibile un aggiornamento di manutenzione per il tuo cluster elastico utilizzando. AWS CLI

Se è disponibile un aggiornamento, puoi scegliere una di queste operazioni:

- Rimanda un'azione di manutenzione attualmente pianificata per la prossima finestra di manutenzione (solo per le patch del sistema operativo).
- Applica immediatamente le operazioni di manutenzione.
- Pianificare le operazioni di manutenzione perché vengano avviate durante la successiva finestra di manutenzione.
- Pianifica l'avvio delle azioni di manutenzione durante la finestra di applicazione selezionata.

La finestra di manutenzione determina l'avvio delle operazioni in sospeso, ma non limita il tempo di esecuzione totale di tali operazioni.

Utilizzate la seguente AWS CLI operazione per determinare quali azioni di manutenzione sono in sospeso. Elenca tutte le azioni di manutenzione in sospeso:

```
aws docdb-elastic list-pending-maintenance-actions
```

L'output di questa operazione è simile al seguente (formato JSON):

```
{
'ResourcePendingMaintenanceActions': [
    {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'ENGINE_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'string',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
    },
],
'NextToken': 'string'
}
```

Ottieni un'eventuale azione di manutenzione in sospeso su un determinato: `resourceArn`

```
aws docdb-elastic get-pending-maintenance-action --resource-arn string-arn
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'ENGINE_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'string',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            }
        ]
    }
}
```

Parametri:

- `ResourceArn`—L'Amazon DocumentDB Amazon Resource Name (ARN) della risorsa a cui si applica l'azione di manutenzione in sospeso.
- `Action`—L'azione di manutenzione in sospeso applicata alla risorsa.

  Valori validi:

  - `ENGINE_UPDATE`
  - `ENGINE_UPGRADE`
  - `SECURITY_UPDATE`
  - `OS_UPDATE`
  - `MASTER_USER_PASSWORD_UPDATE`
- `AutoAppliedAfterDate`—Prima finestra di manutenzione dopo questa data. `NEXT_MAINTENANCE OPT_IN`viene ignorato in questo caso.
- `ForcedApplyDate`—Applicato indipendentemente dalla finestra di manutenzione. `IMMEDIATE OPT_IN`viene ignorato in questo caso.
- `OptInStatus`—Un valore che specifica il tipo di richiesta di opt-in o annulla una richiesta di opt-in. Una richiesta di consenso esplicito di tipo `IMMEDIATE` non può essere annullata.

  Valori validi:

- `IMMEDIATE`—Applica immediatamente l'azione di manutenzione.
- `NEXT_MAINTENANCE`—Applica l'azione di manutenzione durante la successiva finestra di manutenzione della risorsa.
- `APPLY_ON`—Applica l'azione di manutenzione alla data di applicazione specificata indipendentemente dalla prossima finestra di manutenzione per la risorsa.
- `UNDO_OPT_IN`—Annulla qualsiasi richiesta esistente `NEXT_MAINTENANCE` o `APPLY_ON` facoltativa.

- `CurrentApplyDate`—Visualizzato se il tipo di opt-in è. `APPLY_ON`
- `Description`—Una descrizione dell'opzione per l'azione di manutenzione.

Aggiornamenti del motore Elastic Cluster

Con Amazon DocumentDB, puoi scegliere quando applicare le operazioni di manutenzione. Puoi decidere quando Amazon DocumentDB applica gli aggiornamenti utilizzando. AWS CLI

Applica azioni di manutenzione in sospeso:

```
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]
```

Parametri:

- **`--resource-arn`**—L'Amazon DocumentDB Amazon Resource Name (ARN) della risorsa a cui si applica l'azione di manutenzione in sospeso.
- **`--apply-action`**—L'azione di manutenzione in sospeso da applicare a questa risorsa.

  Valori validi:

  - `ENGINE_UPDATE`
  - `ENGINE_UPGRADE`
  - `SECURITY_UPDATE`
  - `OS_UPDATE`
  - `MASTER_USER_PASSWORD_UPDATE`

- **`--opt-in-type`**—Un valore che specifica il tipo di richiesta di opt-in o annulla una richiesta di opt-in. Una richiesta di consenso esplicito di tipo `IMMEDIATE` non può essere annullata.

  Valori validi:

  - `IMMEDIATE`—Applica immediatamente l'azione di manutenzione.
  - `NEXT_MAINTENANCE`—Applica l'azione di manutenzione durante la successiva finestra di manutenzione della risorsa.
  - `APPLY_ON`—Applica l'azione di manutenzione alla data di applicazione specificata indipendentemente dalla prossima finestra di manutenzione per la risorsa.
  - `UNDO_OPT_IN`—Annulla qualsiasi richiesta esistente `NEXT_MAINTENANCE` o `APPLY_ON` facoltativa.
- **`[--apply-on]`**—Obbligatorio se il tipo di opt-in è. `APPLY_ON` Formato: `yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm` (Questa opzione utilizza l'ora UTC. L'ora di inizio può essere in qualsiasi momento futuro, compresa tra un minimo di 30 minuti e un massimo di 14 giorni, oppure la force/apply data dell'azione pendente, a seconda di quale sia la precedente. La finestra temporale dall'inizio alla fine può durare da un minimo di 30 minuti a un massimo di 8 ore.)

L'output di questa operazione è simile al seguente (formato JSON):

```
{
 'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'SECURITY_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'IMMEDIATE',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
 }
}
```

Parametri:

- `ResourceArn`—L'Amazon DocumentDB Amazon Resource Name (ARN) della risorsa a cui si applica l'azione di manutenzione in sospeso.
- `Action`—L'azione di manutenzione in sospeso applicata alla risorsa.

  Valori validi:

  - `ENGINE_UPDATE`
  - `ENGINE_UPGRADE`
  - `SECURITY_UPDATE`
  - `OS_UPDATE`
  - `MASTER_USER_PASSWORD_UPDATE`
- `AutoAppliedAfterDate`—Prima finestra di manutenzione dopo questa data. `NEXT_MAINTENANCE OPT_IN`viene ignorato in questo caso.
- `ForcedApplyDate`—Applicato indipendentemente dalla finestra di manutenzione. `IMMEDIATE OPT_IN`viene ignorato in questo caso.
- `OptInStatus`—Un valore che specifica il tipo di richiesta di opt-in o annulla una richiesta di opt-in. Una richiesta di consenso esplicito di tipo `IMMEDIATE` non può essere annullata.

  Valori validi:

  - `IMMEDIATE`—Applica immediatamente l'azione di manutenzione.
  - `NEXT_MAINTENANCE`—Applica l'azione di manutenzione durante la successiva finestra di manutenzione della risorsa.
  - `APPLY_ON`—Applica l'azione di manutenzione alla data di applicazione specificata indipendentemente dalla prossima finestra di manutenzione per la risorsa.
  - `UNDO_OPT_IN`—Annulla qualsiasi richiesta esistente `NEXT_MAINTENANCE` o `APPLY_ON` facoltativa.
- `CurrentApplyDate`—Visualizzato se il tipo di opt-in è. `APPLY_ON`
- `Description`—Una descrizione dell'opzione per l'azione di manutenzione.

Date di applicazione

Ogni operazione di manutenzione ha una rispettiva data di applicazione che è possibile trovare nella descrizione delle operazioni di manutenzione in attesa. Quando si legge l'output delle azioni di manutenzione in sospeso dal AWS CLI, vengono elencate tre date:

- `CurrentApplyDate`—La data in cui l'azione di manutenzione verrà applicata immediatamente o durante la successiva finestra di manutenzione. Se la manutenzione è facoltativa, questo valore può essere nullo.
- `ForcedApplyDate`—La data in cui la manutenzione verrà applicata automaticamente, indipendentemente dalla finestra di manutenzione.
- `AutoAppliedAfterDate`—La data dopo la quale la manutenzione verrà applicata durante la finestra di manutenzione del cluster.

User-created azioni di manutenzione

In qualità di utente del cluster Amazon DocumentDBElastic, puoi avviare gli aggiornamenti delle configurazioni dei tuoi cluster.

Aggiornamento della password primaria del cluster

```
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
```

Parametri:

- **`--cluster-arn`**—Amazon DocumentDB Amazon Resource Name (ARN) della risorsa a cui verrà applicata l'azione di manutenzione.
- **`[--admin-user-password]`**—La password associata all'utente amministratore.
- **`[--auth-type]`**—Il tipo di autenticazione utilizzato per determinare dove recuperare la password utilizzata per accedere al cluster elastico. I tipi validi sono `PLAIN_TEXT` o. `SECRET_ARN`
- **`[--apply-method]`**—Un valore che specifica il tipo di metodo da applicare. I valori consentiti sono `IMMEDIATE` e `APPLY_ON`. Il valore predefinito è “`IMMEDIATE`”.
- **`[--apply-on]`**—Obbligatorio se lo è. `apply-method` `APPLY_ON` Formato: `yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm` (Questa opzione utilizza l'ora UTC. L'ora di inizio può essere in qualsiasi momento futuro, da un minimo di 30 minuti a un massimo di 14 giorni. La finestra temporale dall'inizio alla fine può durare da un minimo di 30 minuti a un massimo di 8 ore.)

L'output di questa operazione è simile al seguente (formato JSON):

```
{
 'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'MASTER_USER_PASSWORD_UPDATE',
                'OptInStatus': 'APPLY_ON',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
 }
}
```

Modifica delle finestre di manutenzione di Amazon DocumentDB

La finestra di manutenzione deve essere eseguita nel momento di utilizzo più basso e quindi potrebbe essere necessario cambiarla di tanto in tanto. Il cluster elastico non è disponibile durante questo periodo solo se vengono applicate modifiche al sistema (ad esempio una modifica delle operazioni di archiviazione su scala) che richiedono un'interruzione. Non è disponibile solo per il tempo minimo necessario per apportare le modifiche necessarie.

L'impostazione predefinita è una finestra di 30 minuti selezionata a caso da un intervallo di 8 ore per ciascuna regione Amazon Web Services, che si verifica in un giorno della settimana casuale.

Per modificare la finestra di manutenzione, consulta. Modifica delle configurazioni dei cluster elastici

Aggiornamenti del sistema operativo Elastic Cluster

I cluster elastici Amazon DocumentDB richiedono occasionalmente aggiornamenti del sistema operativo. Amazon DocumentDB aggiorna il sistema operativo a una versione più recente per migliorare le prestazioni del database e il livello generale di sicurezza dei clienti. Gli aggiornamenti del sistema operativo non modificano la versione del motore di cluster di un cluster elastico Amazon DocumentDB.

La maggior parte degli aggiornamenti del sistema operativo per i cluster elastici Amazon DocumentDB sono opzionali e non hanno una data fissa per applicarli. Tuttavia, se non applichi questi aggiornamenti per un po' di tempo, potrebbero diventare obbligatori e applicati

automaticamente durante la finestra di manutenzione dei cluster. Questo serve a mantenere il livello di sicurezza del database. Per evitare tempi di inattività imprevisti, applica gli aggiornamenti del sistema operativo al cluster elastico Amazon DocumentDB non appena sono disponibili e imposta la finestra di manutenzione del cluster in un momento adatto alle tue esigenze aziendali.

Crittografia dei dati inattivi per i cluster elastici Amazon DocumentDB

I seguenti argomenti ti aiutano a conoscere, creare e monitorare le chiavi di AWS Key Management Service crittografia per i cluster elastici Amazon DocumentDB:

Argomenti

- In che modo i cluster elastici Amazon DocumentDB utilizzano le sovvenzioni in AWS KMS
- Creazione di una chiave gestita dal cliente
- Monitoraggio delle chiavi di crittografia per i cluster elastici di Amazon DocumentDB
- Ulteriori informazioni

I cluster elastici Amazon DocumentDB si integrano automaticamente con AWS Key Management Service (AWS KMS) per la gestione delle chiavi e utilizzano un metodo noto come crittografia a inviluppo per proteggere i dati. Per ulteriori informazioni sulla crittografia envelope, consulta Crittografia envelope nella Guida per sviluppatori di AWS Key Management Service .

An AWS KMS key è una rappresentazione logica di una chiave. La chiave KMS include metadati, ad esempio l'ID della chiave, la data di creazione, la descrizione e lo stato della chiave. La chiave KMS contiene anche il materiale della chiave utilizzato per crittografare e decrittare i dati. Per ulteriori informazioni sulle chiavi KMS, consulta AWS KMS keys nella Guida per gli sviluppatori di AWS Key Management Service .

I cluster elastici Amazon DocumentDB supportano la crittografia con due tipi di chiavi:

- AWS chiavi di proprietà: i cluster elastici Amazon DocumentDB utilizzano queste chiavi di default per crittografare automaticamente i dati di identificazione personale. Non puoi visualizzare, gestire o utilizzare chiavi di AWS proprietà né controllarne l'uso. Tuttavia, non è necessario effettuare alcuna operazione o modificare programmi per proteggere le chiavi che eseguono la crittografia dei dati. Per ulteriori informazioni, consulta la pagina chiavi di proprietàAWS nella Guida per gli sviluppatori di AWS Key Management Service .

- Customer-managed chiavi: simmetriche, create, AWS KMS keys possedute e gestite dall'utente. Avendo il pieno controllo di questo livello di crittografia, è possibile eseguire operazioni quali:
  - Stabilire e mantenere le policy delle chiavi
  - Stabilire e mantenere le policy e le sovvenzioni IAM
  - Abilitare e disabilitare le policy delle chiavi
  - Ruotare i materiali crittografici delle chiavi
  - Aggiungere tag
  - Creare alias delle chiavi
  - Pianificare l'eliminazione delle chiavi

  Per ulteriori informazioni, consulta Customer managed keys nella Guida per sviluppatori AWS Key Management Service .

> **Important**
>
> È necessario utilizzare una chiave KMS di crittografia simmetrica per crittografare il cluster poiché Amazon DocumentDB supporta solo chiavi KMS di crittografia simmetrica. Non utilizzare una chiave KMS asimmetrica per tentare di crittografare i dati nei tuoi cluster elastici Amazon DocumentDB. Per ulteriori informazioni, consulta Asymmetric keys nella Developer Guide. AWS KMS AWS Key Management Service
>
> Se Amazon DocumentDB non riesce più ad accedere alla chiave di crittografia per un cluster, ad esempio quando l'accesso a una chiave viene revocato, il cluster crittografato passa allo stato terminale. In questo caso, puoi solo ripristinare il cluster da un backup. Per Amazon DocumentDB, i backup sono sempre abilitati per 1 giorno. Inoltre, se disabiliti la chiave per un cluster Amazon DocumentDB crittografato, alla fine perderai l'accesso in lettura e scrittura a quel cluster. Quando Amazon DocumentDB rileva un cluster crittografato da una chiave a cui non ha accesso, imposta il cluster in uno stato terminale. In questo stato, il cluster non è più disponibile e lo stato attuale del database non può essere ripristinato. Per ripristinare il cluster, devi riattivare l'accesso alla chiave di crittografia per Amazon DocumentDB e quindi ripristinare il cluster da un backup.

> **Important**
>
> Non è possibile modificare la chiave KMS per un cluster crittografato dopo averlo già creato. Assicurati di determinare i requisiti della chiave di crittografia prima di creare il cluster elastico crittografato.

In che modo i cluster elastici Amazon DocumentDB utilizzano le sovvenzioni in AWS KMS

I cluster elastici Amazon DocumentDB richiedono una concessione per utilizzare la chiave gestita dal cliente.

Quando crei un cluster crittografato con una chiave gestita dal cliente, i cluster elastici Amazon DocumentDB creano una concessione per tuo conto inviando una richiesta a. `CreateGrant` AWS KMS Le sovvenzioni AWS KMS vengono utilizzate per consentire ai cluster elastici di Amazon DocumentDB di accedere a una chiave KMS in un account cliente.

I cluster elastici Amazon DocumentDB richiedono la concessione per utilizzare la chiave gestita dal cliente per le seguenti operazioni interne:

- Invia `DescribeKey` richieste AWS KMS per verificare che l'ID della chiave simmetrica gestita dal cliente, inserito durante la creazione di un cluster elastico, sia valido.
- Invia `GenerateDataKey` richieste per AWS KMS generare chiavi di dati crittografate dalla chiave gestita dal cliente.
- Invia `Decrypt` richieste per AWS KMS decrittografare le chiavi di dati crittografate in modo che possano essere utilizzate per crittografare i dati.
- Puoi revocare l'accesso alla concessione o rimuovere l'accesso del servizio alla chiave gestita dal cliente in qualsiasi momento. In tal caso, i cluster elastici di Amazon DocumentDB non saranno in grado di accedere a nessuno dei dati crittografati dalla chiave gestita dal cliente, il che influisce sulle operazioni che dipendono da tali dati.

Creazione di una chiave gestita dal cliente

Puoi creare una chiave simmetrica gestita dal cliente utilizzando o l' Console di gestione AWS API. AWS KMS

Creazione simmetrica di chiavi gestita dal cliente

Segui i passaggi per la creazione di una chiave simmetrica gestita dal cliente nella Developer Guide. AWS Key Management Service

Policy della chiave

Le policy della chiave controllano l'accesso alla chiave gestita dal cliente. Ogni chiave gestita dal cliente deve avere esattamente una policy della chiave, che contiene istruzioni che determinano chi può usare la chiave e come la possono usare. Quando crei la chiave gestita dal cliente, è possibile specificare una policy della chiave. Per ulteriori informazioni, consulta le informazioni di accesso con chiave KMS disponibili nella AWS Key Management Service panoramica della Developer Guide. AWS Key Management Service

Per utilizzare la chiave gestita dal cliente con le risorse del cluster elastico di Amazon DocumentDB, le seguenti operazioni API devono essere consentite nella policy chiave:

- `kms:CreateGrant`: aggiunge una concessione a una chiave gestita dal cliente. Le concessioni controllano l'accesso a una chiave KMS specificata, che consente l'accesso alle operazioni di concessione richieste dai cluster elastici di Amazon DocumentDB. Per ulteriori informazioni sull'utilizzo delle sovvenzioni, consulta Grants in nella Developer Guide. AWS KMS AWS Key Management Service
- `kms:DescribeKey`— Fornisce i dettagli delle chiavi gestiti dal cliente per consentire ai cluster elastici di Amazon DocumentDB di convalidare la chiave.
- `kms:Decrypt`— Consente ai cluster elastici di Amazon DocumentDB di utilizzare la chiave di dati crittografata archiviata per accedere ai dati crittografati.
- `kms:GenerateDataKey`— Consente ai cluster elastici di Amazon DocumentDB di generare una chiave dati crittografata e di archiviarla perché la chiave di dati non viene utilizzata immediatamente per la crittografia.

Per ulteriori informazioni, consulta Autorizzazioni per AWS i servizi nelle politiche chiave e Risoluzione dei problemi di accesso con le chiavi nella Guida per gli sviluppatori. AWS Key Management Service

Limitazione dell'accesso alle chiavi gestito dal cliente tramite le politiche IAM

Oltre alle policy chiave KMS, puoi anche limitare le autorizzazioni delle chiavi KMS in una policy IAM.

Esistono diversi modi per rendere la policy IAM più efficace. Ad esempio, per consentire l'utilizzo della chiave gestita dal cliente solo per le richieste che hanno origine nei cluster elastici di Amazon DocumentDB, puoi utilizzare la chiave di condizione con il `kms:ViaService` valore. `docdb-elastic.<region-name>.amazonaws.com`

Per ulteriori informazioni, consultare Consentire agli utenti in altri account di utilizzare una chiave KMS nella Guida per gli sviluppatori di AWS Key Management Service .

Monitoraggio delle chiavi di crittografia per i cluster elastici di Amazon DocumentDB

Quando utilizzi una chiave gestita dal cliente con le tue risorse di cluster elastici Amazon DocumentDB, puoi utilizzare AWS CloudTrail Amazon CloudWatch Logs per tenere traccia delle richieste inviate dai cluster elastici Amazon DocumentDB. AWS KMS

I seguenti esempi sono AWS CloudTrail eventi per `CreateGrant` `GenerateDataKeyWithoutPlainTextDecrypt`, e per `DescribeKey` monitorare AWS KMS key le operazioni chiamate dai cluster elastici Amazon DocumentDB per accedere ai dati crittografati dalla chiave gestita dal cliente:

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"
}
```


Ulteriori informazioni

Le seguenti risorse forniscono ulteriori informazioni sulla crittografia dei dati inattivi:

- Per ulteriori informazioni sui AWS KMS concetti, consulta i concetti AWS Key Management Service di base nella AWS Key Management Service Developer Guide.
- Per ulteriori informazioni sulla AWS KMS sicurezza, consulta le best practice di sicurezza AWS Key Management Service nella AWS Key Management Service Developer Guide.

Service-linked ruoli nei cluster elastici

I cluster elastici Amazon DocumentDB utilizzano ruoli collegati ai servizi AWS Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo unico di ruolo IAM collegato direttamente ai cluster elastici di Amazon DocumentDB. Service-linked i ruoli sono predefiniti dai cluster elastici di Amazon DocumentDB e includono tutte le autorizzazioni richieste dal servizio per chiamare altri servizi per tuo conto. AWS

Un ruolo collegato ai servizi semplifica l'utilizzo dei cluster elastici di Amazon DocumentDB perché non è necessario aggiungere manualmente le autorizzazioni necessarie. I cluster elastici Amazon DocumentDB definiscono le autorizzazioni dei ruoli collegati ai servizi e, se non diversamente definito, solo i cluster elastici Amazon DocumentDB possono assumere i propri ruoli. Le autorizzazioni definite includono la policy di trust e la policy delle autorizzazioni. Una policy delle autorizzazioni specifica non può essere collegata a un'altra entità IAM. È possibile eliminare i ruoli solo dopo aver eliminato le risorse correlate. Questo protegge le risorse dei cluster elastici di Amazon DocumentDB perché non puoi rimuovere inavvertitamente l'autorizzazione di accesso alle risorse.

Per informazioni su altri servizi che supportano i ruoli collegati ai servizi, consulta AWS i servizi che funzionano con IAM e cerca i servizi contrassegnati con Sì nella colonna Ruolo. Service-Linked Scegliere Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato al servizio per tale servizio.

Service-linked autorizzazioni di ruolo per i cluster elastici

I cluster elastici Amazon DocumentDB utilizzano il ruolo collegato ai servizi denominato `AWS ServiceRoleForDocDB-Elastic` per consentire ai cluster elastici Amazon DocumentDB di chiamare i servizi per conto dei tuoi cluster. AWS

A questo ruolo collegato ai servizi è collegata un policy di autorizzazione denominata `AmazonDocDB-ElasticServiceRolePolicy` che concede le autorizzazioni per operare nell'account. La politica delle autorizzazioni dei ruoli consente ai cluster elastici Amazon DocumentDB di completare le seguenti azioni sulle risorse specificate:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/DocDB-Elastic"
                    ]
                }
            }
        }
    ]
}
```

> **Note**
>
> Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato al servizio è necessario configurare le relative autorizzazioni. Se viene visualizzato il seguente messaggio di errore: «Impossibile creare la risorsa. Verify that you have permission to create service linked role. (Verifica di possedere le autorizzazioni necessarie per creare un ruolo collegato ai servizi.) Altrimenti attendi e riprova più tardi.» , assicurati di avere abilitato le seguenti autorizzazioni:

```
{
"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"
        }
    }
}
```

Per ulteriori informazioni, consulta le autorizzazioni dei Service-linked ruoli nella IAM User Guide.

Creazione di un ruolo collegato ai servizi per i cluster elastici di Amazon DocumentDB

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando crei un'istanza DB, i cluster elastici Amazon DocumentDB creano automaticamente il ruolo collegato ai servizi.

Modifica di un ruolo collegato al servizio per i cluster elastici Amazon DocumentDB

I cluster elastici Amazon DocumentDB non consentono di modificare il ruolo collegato al servizio. `AWS ServiceRoleForDocDB-Elastic` Dopo avere creato un ruolo collegato al servizio, non sarà possibile modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato al servizio nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato al servizio per i cluster elastici Amazon DocumentDB

Se non hai più bisogno di utilizzare una funzionalità o un servizio che richiede un ruolo collegato ai servizi, elimina quel ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, prima di poter eliminare il ruolo collegato ai servizi, dovrai eliminare tutti i cluster .

Pulizia di un ruolo collegato ai servizi

Prima di utilizzare IAM per eliminare un ruolo collegato ai servizi, devi innanzitutto verificare che il ruolo non abbia sessioni attive ed eliminare tutte le risorse utilizzate dal ruolo.

Per verificare se il ruolo collegato al servizio ha una sessione attiva nella console IAM:

1. Accedi alla Console di gestione AWS e apri la console IAM.
2. Nel pannello di navigazione della console IAM seleziona Ruoli. Quindi, scegli il nome (non la casella di controllo) del ruolo `AWS ServiceRoleForDocDB-Elastic`.

3. Nella pagina Summary (Riepilogo) per il ruolo selezionato, scegli la scheda Access Advisor (Consulente accessi).

> **Note**
>
> Se non sei sicuro che Amazon DocumentDB elastic clusters stia utilizzando il `AWS ServiceRoleForDocDB-Elastic` ruolo, puoi provare a eliminare il ruolo. Se il servizio utilizza il ruolo, l'eliminazione ha esito negativo ed è possibile visualizzare Regioni AWS dove viene utilizzato il ruolo. Se il ruolo è in uso, prima di poterlo eliminare dovrai attendere il termine della sessione. Non puoi revocare la sessione per un ruolo collegato al servizio. Se desideri rimuovere il `AWS ServiceRoleForDocDB-Elastic` ruolo, devi prima eliminare tutti i cluster.

Eliminazione di tutti i cluster

Per eliminare un cluster nella console Amazon DocumentDB:

1. Accedi Console di gestione AWS e apri la console Amazon DocumentDB.
2. Nel pannello di navigazione scegliere Cluster.
3. Scegli il cluster che desideri eliminare.
4. In Actions (Azioni), scegliere Delete (Elimina).
5. Se ti viene richiesto di creare un'istantanea finale? , scegli Sì o No.
6. Se si sceglie Yes (Sì) nella fase precedente, in Final snapshot name (Nome snapshot finale) immettere il nome dell'ultimo snapshot.
7. Scegli Elimina.

> **Note**
>
> Per eliminare il ruolo collegato ai servizi `AWS ServiceRoleForDocDB-Elastic`, puoi utilizzare la console IAM, l'interfaccia a riga di comando IAM o l'API IAM. Per ulteriori informazioni, consultare Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Quote Amazon DocumentDB

Questo argomento descrive le quote delle risorse e i vincoli di denominazione per Amazon DocumentDB (con compatibilità con MongoDB).

Per alcune funzionalità di gestione, Amazon DocumentDB utilizza una tecnologia operativa condivisa con Amazon Relational Database Service (Amazon RDS) e Amazon Neptune.

Argomenti

- Tipi di istanze supportati
- Regioni supportate
- Quote regionali
- Quote di aggregazione
- Quote del cluster
- Quote delle istanze
- Vincoli per la denominazione
- Vincoli TTL
- Quote elastiche dei cluster
- Quote di frammenti di cluster elastici
- Quote di CPU, memoria, connessione e cursore del cluster elastico per shard

Tipi di istanze supportati

Amazon DocumentDB supporta istanze su richiesta e i seguenti tipi di istanza:

- NVMe-backed:
  - Tipi di istanze R6GD: `db.r6gd.xlarge`,,,,,. `db.r6gd.2xlarge` `db.r6gd.4xlarge` `db.r6gd.8xlarge` `db.r6gd.12xlarge` `db.r6gd.16xlarge`
- Memoria ottimizzata:
  - Tipi di istanza R8G: `db.r8g.large`,,,,`db.r8g.xlarge`,`db.r8g.2xlarge`,`db.r8g.4xlarge`,`db.r8g.8xlarge`. `db.r8g.12xlarge` `db.r8g.16xlarge` `db.r8g.24xlarge` `db.r8g.48xlarge`

  - Tipi di istanza R6G: `db.r6g.large`,,,,`db.r6g.xlarge`,`db.r6g.2xlarge`,`db.r6g.4xlarge`. `db.r6g.8xlarge` `db.r6g.12xlarge` `db.r6g.16xlarge`
  - Tipi di istanza R5: `db.r5.large`,,,`db.r5.xlarge`,`db.r5.2xlarge`,`db.r5.4xlarge`,`db.r5.8xlarge`. `db.r5.12xlarge` `db.r5.16xlarge`
  - Tipi di istanza R4: `db.r4.large`, `db.r4.2xlarge`, `db.r4.4xlarge`, `db.r4.8xlarge` e `db.r4.16xlarge`.
- Prestazioni instabili:
  - Tipi di istanze T4G:. `db.t4g.medium`
  - Tipi di istanze T3:. `db.t3.medium`

Per ulteriori informazioni sui tipi di istanze supportati e sulle loro specifiche, consulta Specifiche della classe di istanza.

Regioni supportate

Amazon DocumentDB è disponibile nei seguenti: Regioni AWS

Nome della regione	Regione	Zone di disponibilità (elaborazione)
Stati Uniti orientali (Ohio)	`us-east-2`	3
Stati Uniti orientali (Virginia settentrionale)	`us-east-1`	6
Stati Uniti occidentali (Oregon)	`us-west-2`	4
Africa (Città del Capo)	`af-south-1`	3
Sud America (San Paolo)	`sa-east-1`	3
Asia Pacifico (Hong Kong)	`ap-east-1`	3
Asia Pacifico (Hyderabad)	`ap-south-2`	3
Asia Pacifico (Malesia)	`ap-southeast-5`	3
Asia Pacifico (Mumbai)	`ap-south-1`	3
Asia Pacifico (Osaka)	`ap-northeast-3`	3
Asia Pacifico (Seul)	`ap-northeast-2`	4
Asia Pacifico (Singapore)	`ap-southeast-1`	3

Nome della regione	Regione	Zone di disponibilità (elaborazione)
Asia Pacifico (Sydney)	ap-southeast-2	3
Asia Pacifico (Giacarta)	ap-southeast-3	3
Asia Pacifico (Melbourne)	ap-southeast-4	3
Asia Pacifico (Thailandia)	ap-southeast-7	3
Asia Pacifico (Tokyo)	ap-northeast-1	3
Canada (Centrale)	ca-central-1	3
Canada occidentale (Calgary)	ca-west-1	3
Regione Cina (Pechino)	cn-north-1	3
Cina (Ningxia)	cn-northwest-1	3
Europa (Francoforte)	eu-central-1	3
Europa (Zurigo)	eu-central-2	3
Europa (Irlanda)	eu-west-1	3
Europa (Londra)	eu-west-2	3
Europa (Milano)	eu-south-1	3
Europa (Parigi)	eu-west-3	3
Europa (Spagna)	eu-south-2	3
Europa (Stoccolma)	eu-north-1	3

Nome della regione	Regione	Zone di disponibilità (elaborazione)
Messico (centrale)	`mx-central-1`	3
Medio Oriente (Emirati Arabi Uniti)	`me-central-1`	3
Israele (Tel Aviv)	`il-central-1`	3
AWS GovCloud (US-West)	`us-gov-west-1`	3
AWS GovCloud (US-East)	`us-gov-east-1`	3

Quote regionali

Per alcune funzionalità di gestione, Amazon DocumentDB utilizza la tecnologia operativa condivisa con Amazon Relational Database Service (Amazon RDS). La tabella seguente contiene le quote regionali condivise tra Amazon DocumentDB e Amazon RDS.

> **Note**
>
> La tecnologia condivisa Amazon RDS descritta sopra si applica solo ai cluster basati su istanze Amazon DocumentDB. I cluster elastici Amazon DocumentDB non condividono la tecnologia con Amazon RDS.

Le seguenti quote si applicano ai cluster basati su istanze Amazon DocumentDB e sono per regione. Account AWS

Risorsa	AWS quota predefinita
Cluster	40
Gruppi di parametri di cluster	50
Cluster per gruppo di parametri	100

Risorsa	AWS quota predefinita
Istanze per gruppo di parametri	300
Abbonamenti a eventi	20
Istanze	40
Snapshot manuali dei cluster	100
Repliche di lettura per cluster	15
Gruppi di sottoreti	50
Sottoreti per gruppo di sottoreti	20
Tag per risorsa	50
Gruppi di sicurezza VPC per istanza	5

Le seguenti quote si applicano ai cluster elastici di Amazon DocumentDB e sono per regione. Account AWS

Risorsa	AWS quota predefinita
Cluster elastici	20
Cluster elastici (vCPU)	1.024
Istantanea manuale dei cluster elastici	20

È possibile utilizzare Service Quotas per richiedere un aumento per una quota, se la quota è regolabile. Alcune richieste vengono risolte automaticamente, mentre altre vengono inviate a Supporto. Puoi monitorare lo stato di una richiesta di aumento della quota inviata a Supporto. Le richieste di aumento delle quote di servizio non ricevono supporto prioritario. Se hai una richiesta urgente, contatta Supporto. Per ulteriori informazioni sulle quote di servizio, vedere Cosa sono le Quote di servizio?

Per richiedere un aumento della quota per Amazon DocumentDB:

1. Aprire la console Quote di servizio in https://console.aws.amazon.com/servicequotas e, se necessario, accedere.
2. Nel pannello di navigazione, scegliere servizi AWS .
3. Seleziona Amazon DocumentDB (con compatibilità con MongoDB) o Amazon DocumentDB Elastic Cluster dall'elenco o digita entrambi nel campo di ricerca.
4. Se la quota è regolabile, è possibile selezionarne il pulsante di opzione o il nome, quindi scegliere Richiedi aumento quota in alto a destra della pagina.
5. In Modifica valore quota, immettere il nuovo valore. Questo valore deve essere maggiore di quello corrente.
6. Scegli Richiedi. Dopo aver risolto la richiesta, il valore della quota applicata per la quota viene impostato sul nuovo valore.
7. Per visualizzare eventuali richieste in sospeso o risolte di recente, scegliere Dashboard dal riquadro di navigazione. Per le richieste in sospeso, scegliere lo stato della richiesta per aprire la ricevuta della richiesta. Lo stato iniziale di una richiesta è `Pending`. Dopo che lo stato è cambiato in`Quota requested`, vedrai il numero del caso con. Supporto Scegli il numero del caso per aprire il ticket della tua richiesta.

Quote di aggregazione

La tabella seguente descrive le quote di aggregazione in Amazon DocumentDB.

Risorsa	Quota
Numero massimo di fasi supportate	500

Quote del cluster

La tabella seguente descrive le quote dei cluster basate su istanze di Amazon DocumentDB.

Risorsa	Quota
Dimensione cluster (somma di tutte le raccolte e gli indici)	256 TiB per la versione 8.0 e successive del motore (128 TiB per le versioni precedenti del motore)
Dimensione della raccolta (la somma di tutte le raccolte non può superare la quota del cluster): non include la dimensione dell'indice	32 TiB
Raccolte per cluster	100.000
Database per cluster	100.000
Dimensione del database (la somma di tutti i database non può superare la quota del cluster)	256 TiB per la versione del motore 8.0 e successive (128 TiB per le versioni precedenti del motore)
Profondità di nidificazione del documento	200 livelli
Dimensioni dei documenti	16 MiB
Dimensione chiave di indice	2.048 byte

Risorsa	Quota
Indici per raccolta	64
Chiavi in un indice composto	32
Numero massimo di operazioni di scrittura in un singolo comando batch	100.000
Numero di utenti per cluster	1000

Quote delle istanze

La tabella seguente descrive le quote di Amazon DocumentDB per istanza.

Amazon DocumentDB 8.0

Tipo di istanza	Memoria dell'istanza (GiB)	Connessioni (tutte)	Quota del cursore	Transazioni aperte	Connessioni (attive)
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

Tipo di istanza	Memoria dell'istanza (GiB)	Connessioni (tutte)	Quota del cursore	Transazioni aperte	Connessioni (attive)
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

* incluso R6GD

Amazon DocumentDB 5.0

Tipo di istanza	Memoria di istanza (GiB)	Connessioni (tutte)	Quota del cursore	Transazioni aperte	Connessioni (attive)
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
R6G.12xlarge*	384	60000	2280	4800	4500
R6G.16xlarge*	512	60000	3040	6400	4500

Tipo di istanza	Memoria di istanza (GiB)	Connessioni (tutte)	Quota del cursore	Transazioni aperte	Connessioni (attive)
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

* incluso R6GD

Amazon DocumentDB 4.0

Tipo di istanza	Memoria di istanza (GiB)	Connessioni (tutte)	Quota del cursore	Transazioni aperte	Connessioni (attive)
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

Tipo di istanza	Memoria di istanza (GiB)	Connessioni (tutte)	Quota del cursore	Transazioni aperte	Connessioni (attive)
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

* incluso R6GD

Amazon DocumentDB 3.6

Tipo di istanza	Memoria di istanza (GiB)	Connessioni (tutte)	Quota del cursore	Transazioni aperte	Connessioni (attive)
T3.medium	4	500	30	N/A	102

Tipo di istanza	Memoria di istanza (GiB)	Connessioni (tutte)	Quota del cursore	Transazioni aperte	Connessioni (attive)
R4.large	15,25	1700	450	N/A	1100
R4.xlarge	30,5	3400	450	N/A	2700
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

Puoi monitorare e segnalare i limiti per istanza utilizzando le seguenti CloudWatch metriche. Per ulteriori informazioni sulle CloudWatch metriche di Amazon DocumentDB, consulta. Monitoraggio di Amazon DocumentDB con CloudWatch

Risorsa	CloudWatch metrica relativa alle quote	CloudWatch metrica di utilizzo (massimo 1 minuto)	CloudWatch metrica di utilizzo
Memoria di istanza	-	-	FreeableMemory

Risorsa	CloudWatch metrica relativa alle quote	CloudWatch metrica di utilizzo (massimo 1 minuto)	CloudWatch metrica di utilizzo
Connessioni (tutte)	DatabaseConnectionsLimit	DatabaseConnectionsMax	DatabaseConnections
Cursori	DatabaseCursorsLimit	DatabaseCursorsMax	DatabaseCursors
Transazioni	TransactionsOpenLimit	TransactionsOpenMax	TransactionsOpen

Vincoli per la denominazione

La tabella seguente descrive i vincoli di denominazione in Amazon DocumentDB.

Risorsa	Quota predefinita
Identificatore del cluster	• La lunghezza è di [1—63] lettere, numeri o trattini. • Il primo carattere deve essere una lettera. • Non può terminare con un trattino o contenere due trattini consecutivi. • Deve essere univoco per tutti i cluster (su Amazon RDS, Amazon Neptune e Amazon DocumentDB) per account, per regione. AWS
Identificatore di istanza	• La lunghezza è di [1—63] lettere, numeri o trattini • Il primo carattere deve essere una lettera • Non può terminare con un trattino o contenere due trattini consecutivi • Deve essere univoco per tutte le istanze (su Amazon RDS, Amazon Neptune e Amazon DocumentDB) per account, per regione. AWS
Nome raccolta: `<col>`	La lunghezza è di [1-255] caratteri (Amazon DocumentDB 5.0 e versioni successive).

Risorsa	Quota predefinita
	La lunghezza è di [1—57] caratteri (Amazon DocumentDB 4.0 e versioni precedenti).
Nome del database: <db>	La lunghezza è di [1—63] caratteri.
Nome completo della collezione: <db>.<col>	La lunghezza è di [3-255] caratteri (Amazon DocumentDB 5.0 e versioni successive). La lunghezza è di [3-120] caratteri (Amazon DocumentDB 4.0 e versioni precedenti).
Nome dell'indice completo: <db>.<col>$<index>	La lunghezza è di [6-511] caratteri (Amazon DocumentDB 5.0 e versioni successive). La lunghezza è di [6-127] caratteri (Amazon DocumentDB 4.0 e versioni precedenti).
Nome indice	La lunghezza è di [1—255] caratteri (Amazon DocumentDB 5.0 e versioni successive). La lunghezza è di [3-63] caratteri per la combinazione <col>$<index> (Amazon DocumentDB 4.0 e versioni precedenti).

Risorsa	Quota predefinita
Password principale	• La lunghezza è di [8-100] caratteri ASCII stampabili. • È possibile utilizzare qualsiasi carattere ASCII stampabile eccetto i seguenti: • **/** (barra) • **"** (virgolette doppie) • **@** (simbolo chiocciola)
Nome utente principale	• La lunghezza è compresa tra 1 e 63 caratteri alfanumerici. • Il primo carattere deve essere una lettera. • Non può essere una parola riservata del motore di database.
Nome del gruppo di parametri	• La lunghezza è compresa tra 1 e 255 caratteri alfanumerici. • Il primo carattere deve essere una lettera. • Non può terminare con un trattino o contenere due trattini consecutivi.

Vincoli TTL

Eliminazioni da un indice TTL non sono garantite all'interno di un determinato periodo di tempo e si basano sul miglior tentativo. Fattori come l'utilizzo di risorse dell'istanza, dimensioni documento e throughput complessivo possono influenzare la tempistica di un'eliminazione TTL.

Quote elastiche dei cluster

La tabella seguente descrive le quote massime per i cluster elastici Amazon DocumentDB.

Risorsa	Quota
Cluster elastici per regione	20
vCPU sommata tra tutti i cluster elastici per regione	1.024
Istantanee manuali dei cluster per regione	20
Frammenti per cluster	32
Storage per cluster (quando i dati sono distribuiti uniformemente tramite shard-key)	4 PiB
Connessioni al cluster	Il valore inferiore di 300.000 o il numero di shard x la quota di connessione associata a vCPU per shard
UnSharded dimensione della raccolta	32 TiB
Dimensione della raccolta frammentata (quando i dati sono distribuiti uniformemente per chiave condivisa)	1 PB
Database per cluster	10.000
UnSharded raccolte per cluster	100.000
Raccolte frammentate per cluster	1000
Utenti per cluster	100
Scrive in un singolo comando batch	100.000
Indici per raccolta	64
Profondità di nidificazione del documento	100 livelli
Dimensioni dei documenti	16MB
Dimensione chiave di indice	2048 byte

Risorsa	Quota
Chiavi in un indice composto	32

Quote di frammenti di cluster elastici

La tabella seguente descrive le quote massime di shard per i cluster elastici Amazon DocumentDB.

Risorsa	Quota
vCPU per istanza di shard	64
Istanze per shard	16
Archiviazione per frammento	128 TiB
Conservazione per raccolta per frammento	32 TiB

Quote di CPU, memoria, connessione e cursore del cluster elastico per shard

La tabella seguente descrive i limiti massimi di CPU, memoria, connessione e cursore negli shard di cluster elastici Amazon DocumentDB.

vCPU per shard	Memoria di istanza (GiB)	Quota di connessione	Quota del cursore
2	16	1700	450
4	32	3500	450
8	64	7100	450
16	128	14200	760
32	256	28400	1520

vCPU per shard	Memoria di istanza (GiB)	Quota di connessione	Quota del cursore
48	384	30000	2280
64	512	30000	3040

Risoluzione dei problemi di Amazon DocumentDB

Le seguenti sezioni forniscono informazioni su come risolvere i problemi che potresti riscontrare quando usi Amazon DocumentDB (con compatibilità con MongoDB).

Argomenti

- Risoluzione dei problemi di connessione
- Indici per la risoluzione dei problemi
- Risoluzione dei problemi relativi alle prestazioni e all'utilizzo delle risorse
- Raccolta dei rifiuti in Amazon DocumentDB
- Codici di errore

Risoluzione dei problemi di connessione

Hai problemi di connessione? Ecco alcuni scenari comuni e come risolverli.

Argomenti

- Impossibile connettersi a un endpoint Amazon DocumentDB
- Test di una connessione a un'istanza Amazon DocumentDB
- Connessione a un endpoint non valido
- La configurazione del driver influisce sul numero di connessioni

Impossibile connettersi a un endpoint Amazon DocumentDB

Quando provi a connetterti ad Amazon DocumentDB, il seguente è uno dei messaggi di errore più comuni che potresti ricevere.

```
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
```

Questo messaggio di errore indica in genere che il tuo cliente (la shell mongo in questo esempio) non può accedere all'endpoint Amazon DocumentDB. Questo potrebbe accadere per vari motivi:

Argomenti

- Connessione da endpoint pubblici
- Cross-Region connessioni
- Connessione da diversi Amazon VPC
- Il gruppo di sicurezza blocca le connessioni in entrata
- Problema relativo alle preferenze di lettura del driver Java Mongo

Connessione da endpoint pubblici

Stai cercando di connetterti a un cluster Amazon DocumentDB direttamente dal tuo laptop o da una macchina di sviluppo locale.

Il tentativo di connessione a un cluster Amazon DocumentDB direttamente da un endpoint pubblico, ad esempio un laptop o una macchina di sviluppo locale, fallirà. Amazon DocumentDB è solo un cloud privato virtuale (VPC) e attualmente non supporta endpoint pubblici. Pertanto, non puoi connetterti direttamente al tuo cluster Amazon DocumentDB dal tuo laptop o da un ambiente di sviluppo locale esterno al tuo VPC.

Per connetterti a un cluster Amazon DocumentDB dall'esterno di un Amazon VPC, puoi utilizzare un tunnel SSH. Per ulteriori informazioni, consulta Connessione a un cluster Amazon DocumentDB dall'esterno di Amazon VPC. Inoltre, se il tuo ambiente di sviluppo si trova in un Amazon VPC diverso, puoi anche utilizzare VPC Peering e connetterti al tuo cluster Amazon DocumentDB da un altro Amazon VPC nella stessa regione o in una regione diversa.

Cross-Region connessioni

Stai tentando di connetterti a un cluster Amazon DocumentDB in un'altra regione.

Se provi a connetterti a un cluster Amazon DocumentDB da un'istanza Amazon EC2 in una regione diversa dalla regione del cluster, ad esempio, provando a connetterti a un cluster nella regione Stati Uniti orientali (Virginia settentrionale) (us-east-1) dalla regione Stati Uniti occidentali (Oregon) (us-west-2), la connessione fallirà.

Per verificare la regione del tuo cluster Amazon DocumentDB, esegui il comando seguente. La regione è nell'endpoint.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].Endpoint'
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
[
    "sample-cluster.node.us-east-1.docdb.amazonaws.com"
]
```

Per verificare la regione dell'istanza EC2, esegui il comando seguente.

```
 aws ec2 describe-instances \
     --query 'Reservations[*].Instances[*].Placement.AvailabilityZone'
```

L'aspetto dell'output di questa operazione è simile al seguente.

```
[
    [
        "us-east-1a"
    ]
]
```

Connessione da diversi Amazon VPC

Stai cercando di connetterti a un cluster Amazon DocumentDB da un VPC diverso da quello Amazon VPC su cui è distribuito il tuo cluster.

Se il cluster Amazon DocumentDB e l'istanza Amazon EC2 sono nello stesso Amazon VPC Regione AWS, ma non nello stesso Amazon VPC, non puoi connetterti direttamente al tuo cluster Amazon DocumentDB a meno che il peering VPC non sia abilitato tra i due Amazon VPC.

Per verificare l'Amazon VPC della tua istanza Amazon DocumentDB, esegui il comando seguente.

```
aws docdb describe-db-instances \
   --db-instance-identifier sample-instance \
   --query 'DBInstances[*].DBSubnetGroup.VpcId'
```

Per verificare Amazon VPC della tua istanza Amazon EC2, esegui il comando seguente.

```
aws ec2 describe-instances \
   --query 'Reservations[*].Instances[*].VpcId'
```

Il gruppo di sicurezza blocca le connessioni in entrata

Stai tentando di connetterti a un cluster Amazon DocumentDB e il gruppo di sicurezza del cluster non consente connessioni in entrata sulla porta del cluster (porta predefinita: 27017).

Supponiamo che il cluster Amazon DocumentDB e l'istanza Amazon EC2 si trovino entrambi nella stessa regione e in Amazon VPC e utilizzino lo stesso gruppo di sicurezza Amazon VPC. Se non riesci a connetterti al tuo cluster Amazon DocumentDB, è probabile che il tuo gruppo di sicurezza (ovvero il firewall) del cluster non consenta le connessioni in entrata sulla porta scelta per il tuo cluster Amazon DocumentDB (la porta predefinita è 27017).

Per verificare la porta per il tuo cluster Amazon DocumentDB, esegui il comando seguente.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[DBClusterIdentifier,Port]'
```

Per ottenere il gruppo di sicurezza Amazon DocumentDB per il tuo cluster, esegui il comando seguente.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[VpcSecurityGroups[*],VpcSecurityGroupId]'
```

Per verificare le regole in entrata per il tuo gruppo di sicurezza, consulta i seguenti argomenti nella documentazione di Amazon EC2:

- Autorizzazione del traffico in entrata per le tue istanze Linux
- Autorizzazione del traffico in entrata per le tue istanze Windows

Problema relativo alle preferenze di lettura del driver Java Mongo

Le preferenze di lettura dei client non vengono rispettate e alcuni client non possono scrivere su Amazon DocumentDB dopo il failover a meno che non si riavviino.

Questo problema, scoperto per la prima volta in Java Mongo Driver 3.7.x, si verifica quando un client stabilisce una connessione ad Amazon DocumentDB utilizzando `MongoClientSettings` e, in particolare, concatenando il metodo. `applyToClusterSettings` Le impostazioni del MongoClient cluster possono essere definite utilizzando alcuni metodi diversi, ad esempio, e. `hosts()` `requiredReplicaSetName()` `mode()`

Quando il client specifica un solo host nel `hosts()` metodo, la modalità è impostata su `ClusterConnectionMode.SINGLE` anziché su `ClusterConnectionMode.MULTIPLE` Ciò fa sì che il client ignori la preferenza di lettura e si connetta solo al server configurato in. `hosts()` Quindi, anche se le impostazioni del client vengono inizializzate come di seguito, tutte le letture andrebbero comunque al primario anziché al secondario.

```
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);
```

Caso di failover

Utilizzando le impostazioni di connessione del client sopra riportate, se si verifica un failover e un aggiornamento ritardato del record DNS per l'endpoint del cluster writer, il client proverebbe comunque a inviare scritture al vecchio writer (ora lettore dopo il failover). Ciò si traduce in un errore sul lato server (non primario) che non viene gestito in modo appropriato dal driver Java (la questione è ancora oggetto di indagine). Pertanto, il client può rimanere in cattivo stato fino al riavvio del server delle applicazioni, ad esempio.

Esistono due soluzioni alternative per questo problema:

- I client che si connettono ad Amazon DocumentDB tramite una stringa di connessione non avranno questo problema, poiché `ClusterConnectionMode` verrà impostato su `MULTIPLE` quando si impostano le preferenze di lettura.

```
MongoClientURI mongoClientURI = new MongoClientURI("mongodb://usr:pass:cluster-
endpoint:27317/test?ssl=false&replicaSet=rs0&readpreference=secondaryPreferred");
MongoClient mongoClient = MongoClients.create(mongoClientURI.getURI());
```

  Oppure usando `MongoClientSettings` builder con il `applyConnectionString` metodo.

```
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);
```

- Impostato esplicitamente su`ClusterConnectionMode`. `MULTIPLE` Ciò è necessario solo quando si utilizza `applyToClusterSettings` e`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);
```

Test di una connessione a un'istanza Amazon DocumentDB

Puoi verificare la connessione a un cluster utilizzando strumenti comuni di Linux o Windows.

Da un terminale Linux o Unix, puoi eseguire il test della connessione immettendo quanto segue (sostituisci `cluster-endpoint` con l'endpoint e `port` con la porta dell'istanza database):

```
nc -zv cluster-endpoint port
```

Di seguito è riportata un'operazione di esempio e il valore restituito:

```
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!
```

Connessione a un endpoint non valido

Quando ci si connette a un cluster Amazon DocumentDB e si utilizza un endpoint del cluster non valido, viene visualizzato un errore simile al seguente.

```
mongo --ssl \
   --host sample-cluster.node.us-east-1.docdb.amazonaws.com:27017 \
   --sslCAFile global-bundle.pem \
   --username <user-name> \
   --password <password>
```

L'output sarà il seguente:

```
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
```

Per ottenere l'endpoint valido per un cluster, esegui il comando seguente:

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[Endpoint,Port]'
```

Per ottenere l'endpoint valido per un'istanza, esegui il comando seguente:

```
aws docdb describe-db-instances \
   --db-instance-identifier sample-instance \
   --query 'DBInstances[*].[Endpoint.Address,Endpoint.Port]'
```

Per ulteriori informazioni, consulta Comprendere gli endpoint di Amazon DocumentDB.

La configurazione del driver influisce sul numero di connessioni

Quando si utilizza il driver del client per connettersi a un cluster Amazon DocumentDB, è importante considerare il parametro di `maxPoolSize` configurazione. L'`maxPoolSize`impostazione determina il numero massimo di connessioni che il driver client manterrà nel suo pool di connessioni.

Indici per la risoluzione dei problemi

I seguenti argomenti spiegano come agire in caso di mancata creazione dell'indice o indice in background.

Argomenti

- La creazione dell'indice non riesce
- L'indice in background genera problemi di latenza e fallisce
- Ingrandimento dell'indice del database

La creazione dell'indice non riesce

Amazon DocumentDB utilizza lo storage locale su un'istanza come parte del processo di creazione dell'indice. È possibile monitorare l'utilizzo del disco utilizzando la FreeLocalStorage CloudWatch metrica (). `CloudWatch -> Metrics -> DocDB -> Instance Metrics` Quando la creazione di un indice consuma tutto il disco locale e ha esito negativo, riceverai un errore. Durante la migrazione dei dati su Amazon DocumentDB, ti consigliamo di creare prima gli indici e poi inserire

i dati. Per ulteriori informazioni sulle strategie di migrazione e sulla creazione di indici, consulta Migrazione e aggiornamento di Amazon DocumentDB la documentazione di Amazon DocumentDB e il blog: Migrazione da MongoDB ad Amazon DocumentDB utilizzando il metodo offline.

Quando crei indici su un cluster esistente, se la creazione dell'indice richiede più tempo del previsto o non riesce, aumenta l'istanza per creare l'indice, quindi ridimensiona di nuovo l'istanza dopo la creazione dell'indice. Amazon DocumentDB consente di scalare rapidamente le dimensioni delle istanze in pochi minuti utilizzando o the. Console di gestione AWS AWS CLI Per ulteriori informazioni, consulta Gestione delle classi di istanze. Grazie ai prezzi delle istanze al secondo, paghi solo per la risorsa che utilizzi al secondo.

L'indice in background genera problemi di latenza e fallisce

Le compilazioni degli indici in background in Amazon DocumentDB non vengono avviate fino al completamento di tutte le query sull'istanza primaria avviate prima dell'avvio della compilazione dell'indice. Se è presente una query di lunga durata, le compilazioni dell'indice in background si bloccheranno fino al termine della query e quindi il completamento può richiedere più tempo del previsto. Questo è vero anche se le raccolte sono vuote.

Le build degli indici in primo piano non presentano lo stesso comportamento di blocco. Invece, le build degli indici in primo piano bloccano in modo esclusivo la raccolta fino al completamento della creazione dell'indice. Per creare indici su una raccolta vuota ed evitare di bloccare le query di lunga durata, utilizzate le build di indici in primo piano.

> **Note**
>
> Amazon DocumentDB consente la creazione di un solo indice di sfondo su una raccolta alla volta. Se le operazioni DDL (Data Definition Language) come `createIndex()` o `dropIndex()` vengono eseguite sulla stessa raccolta durante la creazione di un indice in background, tale creazione non riesce.

Ingrandimento dell'indice del database

Amazon DocumentDB utilizza Multi-Version Concurrency Control (MVCC) per gestire transazioni simultanee. Quando i documenti vengono eliminati o aggiornati, le versioni precedenti rimangono nelle raccolte e negli indici come versioni «morte». Il processo di garbage collection recupera automaticamente lo spazio da queste versioni obsolete per operazioni future.

L'ingrandimento dell'indice si verifica quando gli indici di una raccolta diventano più grandi a causa dell'accumulo di voci di indice morte o obsolete o della frammentazione all'interno delle pagine. La percentuale riportata rappresenta la quantità di spazio nell'indice che può essere utilizzata dalle voci future dell'indice. Questo ingrossamento consuma spazio sia nella cache del buffer che nella memoria. Se desideri rimuovere il gonfiore, dovrai ricostruire gli indici.

Example Esempio

Esegui il comando seguente per determinare lo spazio di archiviazione inutilizzato per il tuo indice:

```
db.coll.aggregate({$indexStats:{}});
```

Il risultato è simile a questo:

```
{
    "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
    }
}
```

È possibile ricostruire gli indici senza tempi di inattività utilizzando il `reIndex` comando, che richiede una scansione dell'intera raccolta. Per informazioni, consulta Manutenzione dell'indice utilizzando ReIndex .

Risoluzione dei problemi relativi alle prestazioni e all'utilizzo delle risorse

Questa sezione fornisce domande e soluzioni per problemi di diagnostica comuni nelle distribuzioni di Amazon DocumentDB. Gli esempi forniti utilizzano la shell Mongo e si riferiscono a una singola istanza. Per trovare un endpoint dell'istanza, consulta Comprendere gli endpoint di Amazon DocumentDB.

Argomenti

- Come posso determinare il numero di operazioni di inserimento, aggiornamento ed eliminazione eseguite sulla mia raccolta tramite l'API Mongo?
- Come si analizzano le prestazioni della cache?
- Come posso trovare e terminare le query bloccate o dalla prolungata esecuzione?
- Come posso visualizzare un piano di interrogazione e ottimizzare un'interrogazione?
- Come posso visualizzare un piano di query nei cluster elastici?
- Come posso creare un elenco di tutte le operazioni in esecuzione su un'istanza?
- Come faccio a sapere se una richiesta sta procedendo?
- Come faccio a determinare il motivo per cui un sistema viene improvvisamente eseguito lentamente?
- Come posso determinare la causa dell'elevato utilizzo della CPU su una o più istanze del cluster?
- Come faccio a determinare i cursori aperti su un'istanza?
- Come posso determinare la versione corrente del motore Amazon DocumentDB?
- Come posso analizzare l'utilizzo degli indici e identificare gli indici inutilizzati?
- Come faccio a identificare gli indici mancanti?
- Come faccio a determinare il numero eccessivo di raccolte di database?
- Riepilogo delle domande utili

Come posso determinare il numero di operazioni di inserimento, aggiornamento ed eliminazione eseguite sulla mia raccolta tramite l'API Mongo?

Per visualizzare il numero di operazioni di inserimento, aggiornamento ed eliminazione eseguite su una determinata raccolta, esegui il comando seguente su quella raccolta:

```
db.collection.stats()
```

L'output di questo comando descrive quanto segue nel relativo opCounters campo:

- num DocsIns - Il numero di documenti inseriti in questa raccolta. Ciò include i documenti inseriti utilizzando i insertMany comandi insert and, nonché i documenti inseriti da un upsert.
- num DocsUpd - Il numero di documenti aggiornati in questa raccolta. Ciò include i documenti aggiornati utilizzando i findAndModify comandi update and.
- num DocsDel - Il numero di documenti eliminati da questa raccolta. Sono inclusi i documenti eliminati utilizzando i findAndModify comandi deleteOne deleteManyremove,, e.
- LastReset: l'ora in cui questi contatori sono stati azzerati l'ultima volta. Le statistiche fornite da questo comando vengono ripristinate quando si esegue starting/stopping il cluster o si ridimensiona up/down l'istanza.

Di seguito db.collection.stats() è riportato un esempio di output dell'esecuzione.

```
{
    "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)
}
```

Questo comando stats deve essere usato quando si visualizzano i contatori specifici della raccolta per le operazioni di inserimento, aggiornamento ed eliminazione tramite l'API Mongo. Un altro modo per visualizzare i contatori delle operazioni specifici della raccolta è abilitare il controllo DML. È possibile visualizzare il numero di operazioni di inserimento, aggiornamento ed eliminazione su tutte le raccolte durante intervalli di tempo di un minuto in. Monitoraggio di Amazon DocumentDB con CloudWatch

Come si analizzano le prestazioni della cache?

L'analisi delle prestazioni della cache può fornire informazioni sull'efficienza del recupero dei dati e sulle prestazioni del sistema e si basa sulla quantità di dati letti dal disco rispetto alla cache. Forniamo statistiche sulla cache sul numero di accessi alla cache (dati letti dalla cache) e di cache mancanti (dati che non vengono trovati nella cache e letti dal disco) per fornire informazioni dettagliate sulle prestazioni della cache. Le statistiche della cache per una raccolta specifica possono essere trovate eseguendo il seguente comando su quella raccolta:

```
db.collection.stats()
```

I valori nel `cacheStats` campo nell'output di questo comando forniscono le statistiche della cache per la raccolta e le statistiche totali della cache per gli indici creati nella raccolta. Queste statistiche sono elencate di seguito:

- **collBlksHit**- Il numero di blocchi letti dalla cache durante le operazioni su questa raccolta.
- **collBlksRead**- Il numero di blocchi letti dal disco (cache mancante) durante le operazioni su questa raccolta.
- **collHitRatio**- La percentuale di accessi alla cache per questa raccolta (`100 * [collBlksHit / (collBlksHit + collBlksRead)]`).
- **idxBlksHit**- Il numero di blocchi letti dalla cache per ogni indice creato in questa raccolta.
- **idxBlksRead**- Il numero di blocchi letti dal disco (cache mancante) per ogni indice creato in questa raccolta.
- **idxHitRatio**- L'hit ratio della cache per gli indici creati su questa raccolta (). `100 * [idxBlksHit / (idxBlksHit + idxBlksRead)]`
- **lastReset**- L'ora in cui queste statistiche sono state ripristinate l'ultima volta. Le statistiche fornite da `db.collection.stats()` vengono ripristinate quando si esegue starting/stopping il cluster o si ridimensiona up/down l'istanza.

Utilizzando il comando è inoltre possibile trovare una suddivisione dei `idxBlksRead` campi `idxBlksHit` e per ciascun indice. `indexStats` Le statistiche della cache specifiche dell'indice possono essere trovate eseguendo il comando seguente:

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

Per ogni indice, le seguenti statistiche sulla cache si trovano nel `cacheStats` campo:

- **blksHit**- Il numero di blocchi letti dalla cache per questo indice.
- **blksRead**- Il numero di blocchi letti dal disco per questo indice.
- **blksHitRatio**- L'hit ratio della cache arrotondato a quattro cifre decimali, calcolato da`100 * [blksHit / (blksHit + blksRead)]`.

Come posso trovare e terminare le query bloccate o dalla prolungata esecuzione?

Le query degli utenti possono essere eseguite lentamente a causa di un piano di query non ottimale o possono essere bloccate a causa di conflitti tra risorse.

Per trovare le query con esecuzione prolungata che rallentano a causa di un piano di query non ottimale o le query bloccate a causa di conflitti tra risorse, utilizza il comando `currentOp`. Puoi

applicare filtri al comando per ridurre l'elenco delle query pertinenti da terminare. È necessario che `opid` sia associato alla query con esecuzione prolungata per poterla terminare.

La query seguente utilizza il comando `currentOp` per elencare tutte le query bloccate o in esecuzione per più di 10 secondi.

```
db.adminCommand({
    aggregate: 1,
    pipeline: [
        {$currentOp: {}},
        {$match:
            {$or: [
                {secs_running: {$gt: 10}},
                {WaitState: {$exists: true}}]}},
        {$project: {_id:0, opid: 1, secs_running: 1}}],
    cursor: {}
});
```

Quindi, puoi restringere la query per trovare l'`opid` di una query in esecuzione per più di 10 secondi e terminarla.

Per trovare e terminare una query in esecuzione da più di 10 secondi

1. Trovare l'`opid` della query.

   ```
   db.adminCommand({
       aggregate: 1,
       pipeline: [
           {$currentOp: {}},
           {$match:
               {$or:
                   [{secs_running: {$gt: 10}},
                    {WaitState: {$exists: true}}]}}],
       cursor: {}
   });
   ```

   L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

   ```
   {
       "waitedMS" : NumberLong(0),
       "cursor" : {
           "firstBatch" : [
   ```

```
            {
                "opid" : 24646,
                "secs_running" : 12
            }
        ],
        "id" : NumberLong(0),
        "ns" : "admin.$cmd"
    },
    "ok" : 1
}
```

2. Terminare la query utilizzando l'operazione `killOp`.

```
db.adminCommand({killOp: 1, op: 24646});
```

Come posso visualizzare un piano di interrogazione e ottimizzare un'interrogazione?

Il rallentamento dell'elaborazione di una query può essere dovuto a un'esecuzione che richiede una scansione completa della raccolta per selezionare i documenti pertinenti. Talvolta, creare indici appropriati consente di accelerare l'esecuzione di una query. Per rilevare questo tipo di scenario e stabilire i campi per cui creare degli indici, puoi usare il comando `explain`.

> **Note**
>
> Amazon DocumentDB emula l'API MongoDB 3.6 su un motore di database appositamente progettato che utilizza un sistema di storage distribuito, tollerante ai guasti e con risoluzione automatica. Di conseguenza, i piani di interrogazione e l'output di possono differire tra Amazon DocumentDB e MongoDB. `explain()` I clienti che desiderano il controllo sul piano di query possono utilizzare l'operatore `$hint` per applicare la selezione di un indice preferito.

Esegui la query che vuoi ottimizzare con il comando `explain` come segue.

```
db.runCommand({explain: {<query document>}})
```

Di seguito è riportato un esempio di operazione.

```
db.runCommand({explain:{
```

```
    aggregate: "sample-document",
    pipeline: [{$match: {x: {$eq: 1}}}],
    cursor: {batchSize: 1}}
});
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "db.test",
        "winningPlan" : {
            "stage" : "COLLSCAN"
        }
    },
    "serverInfo" : {
        "host" : "...",
        "port" : ...,
        "version" : "..."
    },
    "ok" : 1
}
```

L'output precedente indica che la fase `$match` richiede la scansione delle raccolte complete e di verificare che il campo `"x"` di ciascun documento corrisponda a 1. Se la raccolta annovera molti documenti, la sua scansione e, di conseguenza, l'elaborazione completa della query sono molto lente. La presenza di `"COLLSCAN"` nell'output del comando `explain` indica che le prestazioni della query possono migliorare creando indici appropriati.

In questo esempio, la query verifica se il campo `"x"` è uguale a 1 in tutti i documenti. Pertanto, la creazione di un indice sul campo `"x"` consente alla query di evitare la scansione completa della raccolta e di utilizzare l'indice per restituire i documenti pertinenti in meno tempo.

Dopo aver creato un indice sul campo `"x"`, l'output di `explain` è il seguente.

```
{
    "queryPlanner" : {
         "plannerVersion" : 1,
         "namespace" : "db.test",
         "winningPlan" : {
             "stage" : "IXSCAN",
             "indexName" : "x_1",
```

```
            "direction" : "forward"
        }
    },
    "serverInfo" : {
        "host" : "...",
        "port" : ...,
        "version" : "..."
    },
    "ok" : 1
}
```

La creazione di un indice sul campo `"x"` ha permesso alla fase `$match` di avviare una scansione dell'indice e ridurre il numero di documenti per cui valutare il predicato `"x = 1"`.

Per raccolte di piccole dimensioni, il processore di query Amazon DocumentDB può scegliere di non utilizzare un indice se i miglioramenti delle prestazioni sono trascurabili.

Come posso visualizzare un piano di query nei cluster elastici?

Per esaminare un piano di interrogazione nei cluster elastici, usa il `explain` comando. Di seguito è riportato un esempio di `explain` operazione su una query di ricerca destinata a una raccolta frammentata:

```
db.runCommand(
   {
     explain: { find: "cities", filter: {"name": "Seoul"}}
   }
)
```

> **Note**
>
> Amazon DocumentDB emula MongoDB su un motore di database creato appositamente. Di conseguenza, i piani di interrogazione e l'output di `explain()` possono differire tra Amazon DocumentDB e MongoDB. Puoi controllare il piano di query con l'uso dell'`$hint`operatore per imporre la selezione di un indice preferito.

L'output di questa operazione potrebbe essere simile al seguente (formato JSON):

```
{
  "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)
}
```

L'output precedente mostra il piano di interrogazione per l'`find`interrogazione su un cluster a tre frammenti. Ogni shard ha più partizioni di dati che possono avere diversi stadi di input. In questo esempio, viene eseguito un «COLLSCAN» (una scansione della raccolta) su tutte le partizioni prima che i risultati vengano uniti nella fase «PARTITION_MERGE» all'interno di ogni shard. I risultati degli shard vengono quindi uniti nella fase» SHARD_MERGE «prima di essere rispediti al client.

Come posso creare un elenco di tutte le operazioni in esecuzione su un'istanza?

Come utente o utente principale, spesso si desidera elencare tutte le operazioni correnti in esecuzione su un'istanza per scopi di diagnostica e risoluzione dei problemi. Per ulteriori informazioni sulla gestione degli utenti, consulta Gestione degli utenti di Amazon DocumentDB.

Con la `mongo` shell, puoi utilizzare la seguente query per elencare tutte le operazioni in esecuzione su un'istanza Amazon DocumentDB.

```
db.adminCommand({currentOp: 1, $all: 1});
```

La query restituisce l'elenco completo di tutte le query e le attività interne del sistema attualmente operative nell'istanza.

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "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
}
```

Di seguito sono riportati i valori validi per il campo `"desc"`.

- **`INTERNAL`**— Attività interne di sistema come la pulizia del cursore o le attività di pulizia degli utenti obsolete.
- **`TTLMonitor`**— Il thread di monitoraggio Time to Live (TTL). Il suo stato di esecuzione si riflette nel campo `"active"`.
- **`GARBAGE_COLLECTION`**— Il thread interno del Garbage Collector.
- **`CONN`**— La richiesta dell'utente.
- **`CURSOR`**— L'operazione è un cursore inattivo in attesa che l'utente chiami il comando «getMore» per ottenere il successivo batch di risultati. In questo stato, il cursore sta consumando memoria, ma non sta consumando alcun calcolo.

L'output precedente elenca, inoltre, tutte le query degli utenti in esecuzione nel sistema. Ogni query viene eseguita in un contesto formato da un database e una raccolta, detto spazio dei nomi. Lo spazio dei nomi di una query è disponibile nel campo `"ns"`.

A volte è necessario elencare tutte le query degli utenti in esecuzione in un determinato spazio dei nomi. Pertanto, l'output precedente deve essere filtrato in base al campo `"ns"`. Di seguito è riportato un esempio di query per filtrare l'output. La query elenca tutte le query degli utenti attualmente in esecuzione nel database `"db"` e nella raccolta `"test"` (ovvero lo spazio dei nomi `"db.test"`).

```
db.adminCommand({aggregate: 1,
    pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
               {$match: {ns: {$eq: "db.test"}}}],
    cursor: {}
});
```

Come utente principale del sistema, puoi visualizzare le query di tutti gli utenti e anche tutte le attività interne del sistema. Tutti gli altri utenti, invece, possono vedere solo le rispettive query.

Se il numero totale di query e attività di sistema non rientra nelle dimensioni predefinite del cursore di batch, la shell `mongo` genera automaticamente un oggetto iteratole `'it'` che consente di visualizzare il resto dei risultati. Bisogna mantenere in esecuzione il comando `'it'` fino a esaurimento dei risultati.

Come faccio a sapere se una richiesta sta procedendo?

Le query degli utenti possono essere eseguite lentamente a causa di un piano di query non ottimale o possono essere bloccate a causa di conflitti tra risorse. Il debug di questo tipo di query è un processo in più fasi che può richiedere di ripetere più volte una stessa operazione.

Innanzitutto, ai fini del debug, occorre elencare tutte le query bloccate o dall'esecuzione prolungata. La query seguente elenca tutte le query utente che sono state in esecuzione per più di 10 secondi o che sono in attesa di risorse.

```
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: {}
                });
```

Ripeti periodicamente la query precedente per determinare se l'elenco delle query cambia e per identificare le query bloccate o con esecuzione prolungata.

L'eventuale presenza del campo `WaitState` nel documento di output della query d'interesse indica che il motivo alla base del blocco o della lenta esecuzione della query è un conflitto tra le risorse. Il problema relativo alle risorse potrebbe essere dovuto a I/O attività interne del sistema o ad altre domande degli utenti.

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
                "opid" : 201,
                "command" : {
                    "aggregate" : ...
                },
                "secs_running" : 208,
                "WaitState" : "IO"
            }
        ],
        "id" : NumberLong(0),
        "ns" : "admin.$cmd"
    },
    "ok" : 1
}
```

I/O può essere un collo di bottiglia se molte query su raccolte diverse vengono eseguite contemporaneamente sulla stessa istanza o se la dimensione dell'istanza è troppo piccola per il set di dati su cui è in esecuzione la query. Se le query sono di sola lettura, puoi mitigare la situazione precedente separando le query per ogni raccolta tra repliche separate. In caso di aggiornamenti simultanei in raccolte diverse o quando l'istanza è troppo piccola per il set di dati, la soluzione di mitigazione consiste nel ridimensionare l'istanza.

Se il conflitto tra le risorse è causato da query di altri utenti, il campo `"blockedOn"` nel documento di output presenta il valore `"opid"` della query responsabile del problema. Avvaliti della voce `"opid"` e segui la concatenazione dei campi `"WaitState"` e `"blockedOn"` di tutte le query per individuare la query all'inizio della catena.

Se l'attività in testa alla catena è un'operazione interna, la soluzione di mitigazione consiste nel terminare la query ed eseguirla nuovamente in un secondo momento.

Di seguito è riportato un output di esempio in cui la query di ricerca è bloccata su un blocco di raccolta di proprietà di un'altra attività.

```
{
    "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
}
```

Se "WaitState" presenta i valori "Latch", "SystemLock", "BufferLock", "BackgroundActivity" o "Other", il conflitto tra risorse è originato da attività interne di sistema. Se la situazione persiste per un lungo periodo di tempo, l'unica soluzione di mitigazione consiste nel terminare la query ed eseguirla nuovamente in un secondo momento.

Come faccio a determinare il motivo per cui un sistema viene improvvisamente eseguito lentamente?

Di seguito sono elencati alcuni motivi comuni del rallentamento del sistema:

- Eccessivi conflitti delle risorse tra query simultanee
- Il numero di query simultanee attive aumenta nel tempo
- Attività interne di sistema, ad esempio "GARBAGE_COLLECTION"

Per monitorare l'utilizzo del sistema nel tempo, esegui periodicamente la seguente query "currentOp" e trasferisci i risultati su un archivio esterno. La query conta le query e operazioni nei vari spazi dei nomi presenti nel sistema. Puoi analizzare i risultati relativi all'utilizzo del sistema per stabilire il carico del sistema e prendere le decisioni appropriate.

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
                            {$group: {_id: {desc: "$desc", ns: "$ns", WaitState:
 "$WaitState"}, count: {$sum: 1}}}],
                 cursor: {}
                });
```

Questa query restituisce un'aggregazione delle query eseguite nei vari spazi dei nomi e di tutte le attività interne di sistema, nonché il numero univoco degli eventuali stati di attesa per spazio dei nomi.

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{
    "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
}
```

L'output precedente annovera due query dell'utente nello spazio dei nomi `"db.test"` che sono bloccate a causa di un blocco della raccolta, una query nello spazio dei nomi `"admin.$cmd"` e un'attività `"TTLMonitor"` interna.

Se l'output indica il blocco di molte query in stato di attesa, consulta Come posso trovare e terminare le query bloccate o dalla prolungata esecuzione?

Come posso determinare la causa dell'elevato utilizzo della CPU su una o più istanze del cluster?

Le sezioni seguenti possono aiutarti a identificare la causa dell'elevato utilizzo della CPU da parte delle istanze. I risultati possono variare a seconda del carico di lavoro.

- Per determinare il motivo per cui un'istanza viene improvvisamente eseguita lentamente, consulta Come faccio a determinare il motivo per cui un sistema viene improvvisamente eseguito lentamente?
- Per identificare e terminare le query con esecuzione prolungata su una determinata istanza, consulta Come posso trovare e terminare le query bloccate o dalla prolungata esecuzione?
- Per capire se una query sta avanzando, consulta Come faccio a sapere se una richiesta sta procedendo?
- Per determinare il motivo per cui l'esecuzione di una query richiede molto tempo, consulta Come posso visualizzare un piano di interrogazione e ottimizzare un'interrogazione?
- Per tenere traccia delle query con esecuzione prolungata nel corso del tempo, consulta Profilazione delle operazioni di Amazon DocumentDB.

A seconda del motivo dell'elevato utilizzo della CPU da parte delle istanze, può essere utile eseguire una o più delle operazioni indicate di seguito.

- Se l'istanza primaria presenta un elevato utilizzo della CPU diversamente dalle istanze di replica, valuta la possibilità di distribuire il traffico di lettura tra le repliche tramite le impostazioni delle preferenze di lettura del client (ad esempio, `secondaryPreferred`). Per ulteriori informazioni, consulta Connessione ad Amazon DocumentDB come set di repliche.

  L'utilizzo delle repliche per le letture può ottimizzare l'uso delle risorse del cluster consentendo all'istanza primaria di elaborare più traffico di scrittura. Le letture dalle repliche sono consistenti finali.
- Se l'elevato utilizzo della CPU è il risultato del carico di lavoro di scrittura, la modifica delle dimensioni delle istanze del cluster in un tipo di istanza più grande aumenta il numero di core CPU disponibili per il carico di lavoro. Per ulteriori informazioni, consultare Gestione delle istanze di Amazon DocumentDB e Specifiche della classe di istanza.
- Se tutte le istanze del cluster presentano un elevato utilizzo della CPU e il carico di lavoro utilizza le repliche per le letture, l'aggiunta di più repliche al cluster aumenta le risorse disponibili per il traffico

di lettura. Per ulteriori informazioni, consulta Aggiungere un'istanza Amazon DocumentDB a un cluster.

Come faccio a determinare i cursori aperti su un'istanza?

Quando sei connesso a un'istanza Amazon DocumentDB, puoi utilizzare il comando `db.runCommand("listCursors")` per elencare i cursori aperti su quell'istanza. Esiste una quota di massimo 4.560 cursori attivi aperti in qualsiasi momento su una determinata istanza di Amazon DocumentDB, a seconda del tipo di istanza. Chiudi i cursori che non sono più in uso perché i cursori utilizzano risorse su un'istanza. Per ulteriori informazioni, consulta Quote Amazon DocumentDB.

```
db.runCommand("listCursors")
```

Come posso determinare la versione corrente del motore Amazon DocumentDB?

Per determinare la versione attuale del motore Amazon DocumentDB, esegui il comando seguente.

```
db.runCommand({getEngineVersion: 1})
```

L'aspetto dell'output di questa operazione è simile al seguente (formato JSON).

```
{ "engineVersion" : "2.x.x", "ok" : 1 }
```

> **Note**
>
> La versione del motore per Amazon DocumentDB 3.6 è 1.x.x, la versione del motore per Amazon DocumentDB 4.0 è 2.x.x, la versione del motore per Amazon DocumentDB 5.0 è 3.x.x e la versione del motore per Amazon DocumentDB 8.0 è 4.x.x.

Come posso analizzare l'utilizzo degli indici e identificare gli indici inutilizzati?

Per identificare gli indici per una determinata raccolta, eseguire il comando seguente:

```
db.collection.getIndexes()
```

Per analizzare la quantità di indici utilizzati durante le operazioni eseguite sulle raccolte, è possibile utilizzare i comandi `collStats` and`indexStats`. Per visualizzare il numero totale di scansioni eseguite utilizzando gli indici (scansioni degli indici) rispetto al numero di scansioni eseguite senza indice (scansioni delle raccolte), esegui il seguente comando:

```
db.collection.stats()
```

L'output di questo comando include i seguenti valori:

- **`idxScans`**- Il numero di scansioni eseguite su questa raccolta utilizzando un indice.
- **`collScans`**- Il numero di scansioni eseguite su questa raccolta senza utilizzare un indice. Queste scansioni avrebbero comportato l'esame dei documenti della raccolta uno alla volta.
- **`lastReset`**- L'ora in cui questi contatori sono stati azzerati l'ultima volta. Le statistiche fornite da questo comando vengono ripristinate quando si esegue starting/stopping il cluster o si ridimensiona up/down l'istanza.

Un'analisi dettagliata di quanto viene utilizzato ciascun indice è disponibile nell'output del comando seguente. È consigliabile identificare e rimuovere regolarmente gli indici inutilizzati per migliorare le prestazioni e ridurre i costi, in quanto ciò elimina l'elaborazione e lo storage non necessari I/Os per la manutenzione degli indici.

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

L'output di questo comando fornisce i seguenti valori per ogni indice creato nella raccolta:

- **`ops`**- Il numero di operazioni che hanno utilizzato l'indice. Se il carico di lavoro è in esecuzione per un periodo sufficientemente lungo e si è certi che il carico di lavoro è in uno stato costante, un valore `ops` pari a zero indica che l'indice non viene utilizzato affatto.
- **`numDocsRead`**- Il numero di documenti letti durante le operazioni che utilizzano questo indice.
- **`since`**- Il tempo trascorso da quando Amazon DocumentDB ha iniziato a raccogliere statistiche sull'utilizzo dell'indice, che in genere è il valore dall'ultima azione di riavvio o manutenzione del database.
- **`size`**- La dimensione di questo indice in byte.

L'esempio seguente è un esempio di output derivante dall'esecuzione del comando precedente:

```
{
    "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" : ...
    }
}
```

Per determinare la dimensione complessiva dell'indice per una raccolta, eseguire il comando seguente:

```
db.collection.stats()
```

Per eliminare un indice inutilizzato, eseguire il comando seguente:

```
db.collection.dropIndex("indexName")
```

Come faccio a identificare gli indici mancanti?

Puoi utilizzare il profiler Amazon DocumentDB per registrare query lente. Una query che viene visualizzata ripetutamente nel log di query lento potrebbe indicare che è necessario un indice aggiuntivo per migliorare le prestazioni della query.

Puoi identificare le opportunità di creare indici utili cercando query di lunga durata con una o più fasi che eseguono almeno una `COLLSCAN` fase, il che significa che la fase di interrogazione deve leggere tutti i documenti della raccolta per fornire una risposta alla query.

Nell'esempio seguente viene illustrata una query su un insieme di corse in taxi eseguite su una raccolta di grandi dimensioni.

```
db.rides.count({"fare.totalAmount":{$gt:10.0}}))
```

Per eseguire questo esempio, la query doveva eseguire una scansione della raccolta (cioè leggere ogni singolo documento nella raccolta) poiché non vi è alcun indice sul campo `fare.totalAmount`. L'output del profiler Amazon DocumentDB per questa query è simile al seguente:

```
{
    ...
    "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"
}
```

Per velocizzare la query in questo esempio, si desidera creare un indice su `fare.totalAmount`, come illustrato di seguito.

```
db.rides.createIndex( {"fare.totalAmount": 1}, {background: true} )
```

> **Note**
>
> Gli indici creati in primo piano (ovvero se l'opzione `{background:true}` non è stata fornita durante la creazione dell'indice) assumono un blocco di scrittura esclusivo, che impedisce alle applicazioni di scrivere dati nella raccolta fino al completamento della compilazione dell'indice. Tenere presente questo potenziale impatto durante la creazione di indici nei cluster di produzione. Quando crei gli indici, imposta. `{background:true}`

In generale, si desidera creare indici su campi con elevata cardinalità (ad esempio, un numero elevato di valori univoci). La creazione di un indice su un campo con bassa cardinalità può comportare un indice di grandi dimensioni che non viene utilizzato. L'ottimizzatore di query di Amazon DocumentDB considera la dimensione complessiva della raccolta e la selettività degli indici durante la creazione di un piano di query. In alcuni momenti, l'elaboratore di query selezionerà un `COLLSCAN` anche se è presente un indice. Ciò accade quando l'elaboratore di query stima che l'utilizzo dell'indice non produrrà un vantaggio in termini di prestazioni rispetto alla scansione dell'intera raccolta. Se si desidera forzare l'elaboratore di query affinché utilizzi un particolare indice, è possibile ricorrere all'`hint()` come illustrato di seguito.

```
db.collection.find().hint("indexName")
```

Come faccio a determinare il numero eccessivo di raccolte di database?

Il sovraccarico delle raccolte si verifica quando le dimensioni di una raccolta aumentano a causa dell'accumulo di documenti morti o obsoleti o della frammentazione all'interno delle pagine del database. La percentuale riportata rappresenta la quantità di spazio per i documenti che possono essere utilizzati dai documenti futuri. Questo ingombro consuma spazio sia nella cache del buffer che nella memoria. Per eliminare il sovraccarico, le raccolte devono essere ricaricate tramite dump/restore o utilizzando un loop-back di migrazione e switchover durante una finestra di manutenzione.

Example Esempio

Esegui il comando seguente per determinare lo spazio di archiviazione inutilizzato per la tua raccolta:

```
db.runCommand({collStats:'coll'})
```

Il risultato è simile a questo:

```
{
        "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)
}
```

Riepilogo delle domande utili

Le seguenti domande possono essere utili per monitorare le prestazioni e l'utilizzo delle risorse in Amazon DocumentDB.

- Utilizza il seguente comando per visualizzare le statistiche su una raccolta specifica, inclusi contatori delle operazioni, statistiche sulla cache, statistiche sugli accessi e statistiche sulle dimensioni:

```
db.collection.stats()
```

- Utilizzate il comando seguente per visualizzare le statistiche su ogni indice creato in una raccolta, tra cui la dimensione dell'indice, le statistiche della cache specifiche dell'indice e le statistiche sull'utilizzo dell'indice:

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

- Utilizza la seguente query per elencare tutte le attività.

```
db.adminCommand({currentOp: 1, $all: 1});
```

- Il codice seguente elenca tutte le query bloccate o con esecuzione prolungata.

```
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: {}
                });
```

- Il codice seguente termina una query.

```
db.adminCommand({killOp: 1, op: <opid of running or blocked query>});
```

- Utilizza il codice seguente per ottenere una visualizzazione aggregata dello stato del sistema.

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
                            {$group: {_id: {desc: "$desc", ns: "$ns", WaitState:
 "$WaitState"}, count: {$sum: 1}}}],
                 cursor: {}
                });
```

Raccolta dei rifiuti in Amazon DocumentDB

Amazon DocumentDB implementa un'architettura di database MVCC (Multi-version Concurrency Control) che crea nuove versioni di documenti e voci di indice per ogni operazione di aggiornamento. Questa architettura consente l'isolamento delle transazioni, impedendo che le modifiche apportate a una transazione vengano visualizzate in un'altra.

Argomenti

- Informazioni sulla Garbage Collection in Amazon DocumentDB
- Processo di raccolta dei rifiuti
- Architettura di archiviazione e archiviazione estesa
- Monitoraggio della raccolta dei rifiuti
- Esempio di output CollStats
- Domande frequenti

Informazioni sulla Garbage Collection in Amazon DocumentDB

La Garbage collection (GC) è un processo automatizzato in background che mantiene prestazioni e disponibilità di sistema ottimali in Amazon DocumentDB. Come molti database moderni, l'architettura MVCC di Amazon DocumentDB crea nuove versioni di documenti e indici ad ogni aggiornamento. Ogni operazione di scrittura utilizza un ID MVCC univoco da un contatore finito. Questi ID identificano a quale transazione appartiene una versione del documento e se è stata salvata o annullata. Nel tempo, queste vecchie versioni e i relativi ID MVCC si accumulano e richiedono una pulizia per evitare un peggioramento delle prestazioni.

Funzioni della raccolta dei rifiuti

Garbage collector svolge tre funzioni essenziali:

- Recupera spazio di archiviazione: rimuove le versioni obsolete di documenti e indici che non sono più necessarie per le query attive, liberando spazio per future operazioni di scrittura.
- Impedisce l'overflow degli ID MVCC: previene l'overflow degli ID MVCC gestendo il contatore finito degli ID MVCC. Senza questa gestione, il contatore alla fine raggiungerebbe il limite, costringendo il database a una modalità temporanea di sola lettura fino a quando gli ID non vengono riciclati.
- Mantiene le prestazioni delle query: mantiene prestazioni ottimali delle query eliminando le versioni inattive dei documenti che altrimenti si accumulerebbero e rallenterebbero l'elaborazione delle query.

Processo di raccolta dei rifiuti

Il processo GC opera per raccolta e può avere più processi in esecuzione contemporaneamente su raccolte diverse. Il processo è costituito da quattro fasi sequenziali:

1. Identificazione: il sistema identifica le versioni del documento e dell'indice non più referenziate da transazioni o interrogazioni attive.
2. Caricamento in memoria: i vecchi documenti e le voci dell'indice vengono caricati in memoria se non sono già presenti.
3. Eliminazione: le versioni obsolete vengono eliminate definitivamente per recuperare spazio di archiviazione.
4. Riciclaggio degli ID MVCC: il sistema ricicla gli ID MVCC dalle versioni eliminate per nuove operazioni.

Quando Garbage Collection completa l'elaborazione delle vecchie versioni dei documenti, rimuove gli ID MVCC più vecchi dal sistema. Questa pulizia è fondamentale per prevenire l'overflow degli ID MVCC riciclando gli ID MVCC e rendendoli disponibili per nuove operazioni di scrittura nel cluster. Senza questo processo di riciclo, il sistema finirebbe per esaurire il numero limitato di ID MVCC e entrare in uno stato di sola lettura.

Pianificazione della raccolta dei rifiuti

La raccolta dei rifiuti viene eseguita automaticamente in background a intervalli periodici. La tempistica e la frequenza si regolano dinamicamente in base al carico del sistema, alle risorse disponibili, al volume di scrittura e ai livelli di consumo dell'ID MVCC. Durante un'elevata attività di scrittura, il processo GC viene eseguito più frequentemente per gestire l'aumento del numero di versioni dei documenti.

Architettura di archiviazione e archiviazione estesa

Amazon DocumentDB utilizza una sofisticata architettura di storage che separa l'archiviazione dei documenti in due segmenti distinti:

Segmento di storage di base

Il segmento di archiviazione di base contiene i dati e i metadati del documento principale. Questo segmento memorizza:

- Contenuto del documento che rientra nelle dimensioni standard della pagina (8 KB).
- Metadati del documento e informazioni sulla struttura.
- Indici primari e relative voci.
- Collection-level statistiche e configurazione.

Segmento di archiviazione esteso

Il segmento di archiviazione esteso utilizza un archivio di oggetti di documenti di grandi dimensioni specializzato progettato per gestire documenti che superano le dimensioni standard della pagina di archiviazione. Questo segmento offre:

- Gestione efficiente dei documenti di grandi dimensioni: i documenti più grandi della soglia di archiviazione di base vengono automaticamente spostati nel segmento di archiviazione esteso.
- Layout di archiviazione ottimizzato: il segmento utilizza un diverso formato di archiviazione ottimizzato per oggetti di grandi dimensioni, che riduce la frammentazione e migliora i modelli di accesso.
- Raccolta indipendente dei rifiuti: il segmento di archiviazione esteso ha un proprio processo di raccolta dei rifiuti che può essere eseguito indipendentemente dalla pulizia dello storage di base.
- Accesso trasparente: le applicazioni accedono senza problemi a documenti di grandi dimensioni senza dover sapere quale segmento di archiviazione contiene i dati.

Il segmento di archiviazione esteso è particolarmente utile per:

- Raccolte con documenti contenenti matrici incorporate di grandi dimensioni.
- Documenti con strutture annidate estese.
- Raccolte che memorizzano dati binari o campi di testo di grandi dimensioni.

- Applicazioni con documenti di dimensioni diverse in cui alcuni documenti superano notevolmente le dimensioni medie.

Monitoraggio della raccolta dei rifiuti

Metriche a livello di cluster

`AvailableMVCCIds`

- Luogo: Amazon CloudWatch
- Descrizione: un contatore che mostra il numero di operazioni di scrittura rimanenti disponibili a partire da un limite massimo di 1,8 miliardi. Quando questo contatore raggiunge lo zero, il cluster entra in modalità di sola lettura fino a quando gli ID non vengono recuperati e riciclati. Il contatore diminuisce a ogni operazione di scrittura e aumenta man mano che la Garbage Collection ricicla i vecchi ID MVCC.
- Raccomandazione: imposta un allarme quando il valore scende al di sotto di 900 milioni su Amazon DocumentDB 5.0 e versioni successive o inferiore a 1,3 miliardi su Amazon DocumentDB 3.6 e 4.0. Questo avviso precoce consente di adottare le misure consigliate discusse più avanti.

`LongestActiveGCRuntime`

- Ubicazione: Amazon CloudWatch
- Descrizione: durata in secondi del processo di raccolta dei rifiuti attivo più lungo. Si aggiorna ogni minuto e tiene traccia solo delle operazioni attive, esclusi i processi che vengono completati entro la finestra di un minuto.
- Raccomandazione: confrontalo con i dati `gcRuntimeStats` storici per identificare comportamenti anomali nella raccolta dei rifiuti, ad esempio tempi di esecuzione prolungati durante le eliminazioni di massa.

Metriche relative al livello di raccolta

`MVCCIDStats: MVCCIdScale`

- Posizione: comando CollStats del database

- Descrizione: misura l'età dell'ID MVCC su una scala da 0 a 1, dove 1 indica l'età massima prima che un cluster entri in uno stato di sola lettura. Usa questa metrica insieme `AvailableMVCCIds` per identificare le raccolte contenenti gli ID MVCC più vecchi che stanno invecchiando il cluster.
- Raccomandazione: mantieni valori inferiori a 0,3 per ogni raccolta.

`gcRuntimeStats`

- Posizione: comando CollStats del database
- Descrizione: fornisce una cronologia di due mesi delle metriche di Garbage Collection, tra cui le esecuzioni totali, la durata media e la durata massima. Include solo le operazioni di raccolta dei rifiuti che durano più di cinque minuti per garantire statistiche significative.

`storageSizeStats`

- Posizione: comando CollStats del database
- Descrizione: fornisce una suddivisione dettagliata dell'utilizzo dello storage tra diversi segmenti di storage:
  - `storageSegmentBase`— Archiviazione utilizzata dal segmento di archiviazione di base per i documenti standard
  - `storageSegmentExtended`— Archiviazione utilizzata dal segmento di archiviazione esteso per documenti di grandi dimensioni
- Utilizzo: aiuta a identificare le raccolte con una notevole quantità di documenti di grandi dimensioni e a comprendere i modelli di distribuzione dello storage.

`unusedStorageSize`(livello di raccolta)

- Posizione: comando CollStats del database
- Descrizione: stima lo spazio di archiviazione inutilizzato in una raccolta in base a statistiche campionate. Include lo spazio dei documenti eliminati e dei segmenti vuoti. La metrica fornisce sia i totali combinati che le suddivisioni per segmento:
  - Combinato e in tutti i segmenti di storage `unusedBytes` `unusedPercent`
  - `storageSegmentBase`— Spazio inutilizzato in particolare nel segmento di archiviazione di base

- `storageSegmentExtended`— Spazio inutilizzato specificamente nel segmento di archiviazione esteso

`documentFragmentStats`

- Posizione: comando CollStats del database
- Descrizione: fornisce informazioni dettagliate sui frammenti di documenti e sui dati non più disponibili all'interno delle raccolte. I frammenti di documento rappresentano le unità di archiviazione interne utilizzate dal motore di database, mentre i frammenti morti indicano dati che non sono più accessibili ma non sono ancora stati recuperati. Questa metrica include:
  - `totalDocFragmentsCount`— Numero totale di frammenti di documento nella raccolta
  - `deadDocFragmentsCount`— Numero di frammenti contenenti dati morti (inaccessibili)
  - `deadDocFragmentsPercent`— Percentuale di frammenti che contengono dati non validi
  - `deadDocFragmentBytes`— Stima dei byte consumati dai frammenti di documento non funzionanti
  - Per-segment suddivisione per e `storageSegmentBase` `storageSegmentExtended`
- Utilizzo: monitora questa metrica per comprendere l'efficacia della raccolta dei rifiuti e identificare le raccolte che potrebbero trarre vantaggio dalle operazioni di manutenzione. Le alte percentuali di frammenti morti indicano che la raccolta dei rifiuti potrebbe essere in ritardo o che la raccolta trarrebbe vantaggio dall'ottimizzazione.

Metriche a livello di indice

`unusedStorageSize`(livello di indice)

- Posizione: comando Database IndexStats
- Descrizione: stima lo spazio di archiviazione inutilizzato in un indice in base a statistiche campionate. Include lo spazio contenuto nelle voci dell'indice obsolete e nei segmenti vuoti.
- Raccomandazione: utilizzate il `reIndex` comando per ricostruire gli indici senza tempi di inattività e recuperare spazio inutilizzato. Per ulteriori informazioni, fare riferimento a Gestione degli indici.

Esempio di output CollStats

L'esempio seguente mostra un `collStats` output tipico con metriche di garbage collection e storage:

```
{
    "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)
}
```

Domande frequenti

Come posso identificare se la garbage collection non funziona in modo efficiente?

Monitora questi segnali di avvertimento che indicano una raccolta dei rifiuti inefficiente:

- Eccessivo aumento della raccolta: `unusedStorageSize` metriche in costante aumento durante scritture pesanti o eliminazioni in blocco, specialmente con indici di grandi dimensioni.
- Elevata percentuale di frammenti morti: `documentFragmentStats` mostra valori costantemente elevati (superiori al 10-15%). `deadDocFragmentsPercent`
- Latenza delle query ridotta: maggiore latenza delle query a causa dell'accumulo di documenti inattivi.
- Durata GC estesa: le operazioni di raccolta dei rifiuti richiedono più tempo rispetto alle medie storiche in. `gcRuntimeStats`
- Elaborazione GC elevata: `LongestActiveGCRuntime` valore elevato che indica che il garbage collector non è in grado di tenere il passo con le richieste di sistema.

La garbage collection influisce sulle prestazioni del mio database?

In condizioni normali, la garbage collection ha un impatto minimo sulle prestazioni. Tuttavia, quando la raccolta dei rifiuti non funziona, potresti riscontrare:

- Aumento dei costi di archiviazione dovuti all'accumulo di documenti obsoleti.
- Prestazioni delle query più lente a causa di voci di indice obsolete.
- Modalità temporanea di sola lettura se gli ID MVCC sono esauriti.
- Maggiore utilizzo delle risorse durante le operazioni di raccolta intensive, specialmente su istanze più piccole.
- Efficienza ridotta nelle operazioni estese dei segmenti di archiviazione per documenti di grandi dimensioni.

Posso attivare manualmente la raccolta dei rifiuti?

No, la garbage collection in Amazon DocumentDB non può essere attivata manualmente. Il sistema gestisce automaticamente la raccolta dei rifiuti come parte delle sue operazioni di manutenzione interne.

Quali allarmi devo impostare come best practice operativa?

Imposta il monitoraggio sia a livello di cluster che di raccolta per garantire prestazioni ottimali del tuo sistema Amazon DocumentDB.

Per il monitoraggio a livello di cluster, inizia creando un CloudWatch allarme Amazon per la `AvailableMVCCIds` metrica con una soglia di 900 milioni su Amazon DocumentDB 5.0 e versioni successive o 1,3 miliardi su Amazon DocumentDB 3.6 e 4.0. Questo ti dà il tempo sufficiente per intervenire prima che la metrica raggiunga lo zero, dopodiché il cluster entrerà in modalità di sola lettura. Tieni presente che questa metrica può variare in base ai tuoi modelli di utilizzo specifici: alcuni clienti la vedono scendere al di sotto della soglia e poi recuperarla oltre 1,5 miliardi quando la Garbage Collection completa il suo lavoro.

È inoltre importante monitorare la metrica tramite Amazon. `LongestActiveGCRuntime` CloudWatch Questa metrica, insieme `gcRuntimeStats` a, ti aiuta a capire l'efficienza della Garbage Collection in tutto il tuo sistema.

Per il monitoraggio a livello di raccolta, concentrati su queste metriche chiave:

- `MVCCIdScale`— Fai attenzione ai valori crescenti che suggeriscono che gli ID MVCC stanno invecchiando e potrebbero richiedere attenzione.
- `gcRuntimeStats`— Identifica i processi di raccolta dei rifiuti che richiedono tempi insolitamente lunghi o che si estendono su più giorni.
- `documentFragmentStats`— `deadDocFragmentsPercent` Valori di monitoraggio: percentuali costantemente elevate (superiori al 10-15%) possono indicare che la raccolta dei rifiuti è in ritardo.
- `storageSizeStats`e `unusedStorageSize` — Tieni traccia dei modelli di utilizzo dello storage e identifica le raccolte con una significativa quantità di spazio inutilizzato in entrambi i segmenti di archiviazione.

Le raccolte con frequenti operazioni di scrittura richiedono un'attenzione particolare, in quanto generano più lavoro per il garbage collector. Controlla queste metriche più frequentemente per le raccolte con un'intensa attività di scrittura per assicurarti che Garbage Collection sia al passo con il tuo carico di lavoro.

Tieni presente che questi consigli di monitoraggio servono come punto di partenza. Man mano che acquisisci maggiore familiarità con il comportamento del tuo sistema, potresti voler modificare queste soglie per adattarle meglio ai tuoi modelli e requisiti di utilizzo specifici.

Cosa devo fare se i miei **`MVCCID disponibili scendono al di sotto della soglia`** di allarme?

Se la `AvailableMVCCIds` metrica scende al di sotto della soglia di allarme, agisci immediatamente per evitare che il cluster entri in modalità di sola lettura. Innanzitutto, aumenta le dimensioni dell'istanza per fornire al Garbage Collector più risorse di elaborazione. Ciò consente all'applicazione di continuare le normali operazioni fornendo al garbage collector la potenza aggiuntiva di cui ha bisogno per recuperare il ritardo.

Se la scalabilità da sola non migliora la situazione, valuta la possibilità di ridurre le operazioni di scrittura. Utilizzate la `MVCCIdScale` metrica per identificare quali raccolte specifiche contengono ID MVCC precedenti che richiedono attenzione. Inoltre, esegui `documentFragmentStats` il monitoraggio per identificare le raccolte con percentuali elevate di frammenti morti che potrebbero contribuire all'inefficienza della raccolta dei rifiuti.

Una volta identificate queste raccolte, potrebbe essere necessario ridurne temporaneamente le operazioni di scrittura per consentire a Garbage Collection di recuperare il ritardo accumulato. Durante il periodo di recupero, monitora attentamente la `AvailableMVCCIds` metrica per assicurarti che le tue azioni abbiano l'effetto desiderato. Il cluster è considerato integro quando il `AvailableMVCCIds` valore torna a 1,5 miliardi o più.

Ricorda che questi passaggi sono misure preventive per aiutare il sistema a ripristinarsi prima che raggiunga uno stato critico. Quanto prima si interviene dopo aver visto la metrica scendere al di sotto della soglia, tanto più è probabile che si eviti qualsiasi impatto sulle operazioni di scrittura.

Le operazioni in batch influiscono sul tasso di consumo dell'ID MVCC?

Sì. Poiché un batch eseguito all'interno di una singola transazione atomica consuma solo un ID MVCC, il raggruppamento delle scritture in batch riduce la velocità di utilizzo degli ID MVCC. Come best practice, utilizzare `insertMany` e, `updateMany` ove possibile, ridurre al minimo il consumo di ID MVCC e ridurre il carico sulla garbage collection.

Perché gli **`MVCCID disponibili non vengono ripristinati fino al raggiungimento`** di una determinata soglia?

Per un carico di lavoro dominante in `AvailableMVCCIds` fase di scrittura che non ha ancora utilizzato un numero sufficiente di ID MVCC per attivare il ciclo di recupero della Garbage Collection, è previsto temporaneamente un calo costante e senza ripristino. La garbage collection si attiva

quando viene soddisfatta una delle seguenti condizioni: una determinata parte dei documenti all'interno di una raccolta diventa obsoleta, una determinata porzione di nuovi documenti viene inserita o scende al di sotto della soglia di recupero. `AvailableMVCCIds`

Fino a quando non `AvailableMVCCIds` scende al di sotto della soglia di recupero, la bonifica forzata non si verifica. Una volta effettuata, la bonifica si attiva e verrà ripristinata. `AvailableMVCCIds` Tuttavia, le transazioni di lunga durata e i cursori aperti possono fissare l'orizzonte di recupero e impedirne il recupero anche dopo il raggiungimento della soglia.

Codici di errore

Quando un comando fallisce, ricevi un errore da Amazon DocumentDB (con compatibilità MongoDB) che contiene un errore numerico `code` e un messaggio descrittivo (). `errmsg` Utilizza la seguente tabella per identificare i codici di errore che Amazon DocumentDB 8.0 potrebbe restituire alla tua applicazione client. Se un codice di errore corrisponde a una singola condizione, la descrizione corrisponde al testo del messaggio restituito dal server. I codici che coprono più condizioni hanno una descrizione generale.

> **Important**
>
> Non creare una logica applicativa che dipenda da codici di errore numerici specifici perché Amazon DocumentDB potrebbe modificare i codici di errore tra le versioni principali.

Codice di errore	Description
1	Articolo non trovato
2	L'argomento, l'opzione o il valore del campo di un comando non sono validi o non sono supportati.
4	Manca un campo obbligatorio o un valore di configurazione non è formattato correttamente.
5	La definizione di una vista fa riferimento a se stessa, crea un ciclo o si è verificato un errore nel file system.

Codice di errore	Description
9	Un comando, un campo, un suggerimento o un token di curriculum non può essere analizzato o ha un formato non valido.
10	La raccolta di destinazione non è stata trovata o non è stata specificata alcuna raccolta.
11	Non è stato possibile trovare un utente referenziato o la raccolta di destinazione esiste già.
13	Non sei autorizzato a eseguire questo comando o la tua autenticazione non è valida.
14	Un campo o un valore ha un tipo di dati errato.
15	Un valore ha superato l'intervallo consentito durante la valutazione.
16	Il batch di scrittura è vuoto, non valido o supera il numero di operazioni consentit o.
17	Non tutti gli oggetti sono dello stesso tipo
18	Autenticazione non riuscita perché le credenziali non sono valide o l'utente non esiste.
19	Operazione primitiva di elaborazione dei dati non trovata o tipo di primitivo non trovato
20	L'operazione richiesta non è consentita in questo contesto.
26	Lo spazio dei nomi o la raccolta di destinazione non esiste.
27	L'indice richiesto non esiste o non è disponibile un indice adatto.
37	Un argomento della pipeline o della fase di aggregazione è mancante, vuoto o non valido.
38	Nome del database non trovato

Codice di errore	Description
39	L'operazione è stata annullata perché ha superato MaxTimeMS, è durata troppo a lungo o ha superato la memoria disponibile.
40	Un aggiornamento è in conflitto con un altro campo oppure le scritture vengono limitate o disabilitate dalla Garbage Collection.
41	JavaScript errore
43	Il cursore specificato non esiste più o è scaduto.
44	Indice dell'array fuori dai limiti
48	Esiste già una raccolta, una vista o uno spazio dei nomi con quel nome.
51	Operatore di aggiornamento illegale
52	Il nome di un campo utilizza un prefisso '$' riservato che non è valido per l'archivi azione.
53	Il campo _id ha un tipo non valido o non supportato, ad esempio un array o un'espressione regolare.
54	non può dedurre i campi di interrogazione da impostare
55	Transazione già effettuata
56	Il nome di un campo nella richiesta è vuoto, il che non è consentito.
57	Transazione invalidata da un'altra transazione
58	Valore del campo troppo lungo
59	Il comando richiesto è sconosciuto o non supportato.
61	Raccolta non valida
62	Impossibile apportare modifiche alla raccolta

Codice di errore	Description
65	si sono verificati errori di batch op
66	Un aggiornamento ha tentato di modificare il campo _id immutabile.
67	L'indice non può essere creato a causa di una specifica non valida o non supportata.
68	Esiste già un indice con proprietà in conflitto.
70	il campo utilizzato con $snippet non è configurato come colonna snippet
71	L'indice esiste già con diverse opzioni
72	Una o più opzioni specificate per il comando o l'indice non sono valide.
73	Il nome dello spazio dei nomi, del database o della raccolta non è valido o non è consentito per questa operazione.
74	La raccolta non ha un indice di testo definito.
75	Un indice di testo è definito per questa raccolta, ma non per questo campo.
77	Impossibile utilizzare l'istantanea con ordinamento
78	Documento non trovato
80	La query supera la dimensione massima consentita.
82	L'intervallo di tempo ha una formattazione non valida. <number>Formato valido: '1 m'o '3 y' '//' h/d w/s m/y
85	Esiste già un indice con questo nome ma con opzioni diverse.
86	L'indice richiesto è in conflitto con un indice esistente con lo stesso nome.
87	L'ordine delle chiavi $sort deve essere 1 (in ordine crescente) o -1 (in ordine decrescente)

Codice di errore	Description
92	Spazio insufficiente per il funzionamento
93	L'utente esiste già
94	Il privilegio della raccolta, del cluster o della risorsa di destinazione non è valido o non è disponibile.
95	Il database non esiste
96	Il parametro 'WriteConcern' non è valido
97	La password non può essere vuota
99	Ruolo o privilegio non definito
102	Impossibile aggiungere un ruolo già integrato
103	Non è stato possibile estrarre un ruolo valido dalla raccolta.
105	L'utente non è valido o la libreria di autorizzazione non può essere inizializzata.
107	Al database non possono essere assegnati ruoli da un altro database
111	_id deve essere specificato esattamente quando si fornisce il documento di aggiornamento
112	Scrivi conflitto
113	Il ruolo specificato non esiste.
114	Il ruolo esiste già
115	AnyResource deve essere vero quando specificato
116	le risorse devono avere db e collection set, cluster set o AnyResource set
117	L'operazione è in conflitto con un'altra operazione in corso, ad esempio una transazione attiva.

Codice di errore	Description
120	«ShowPrivileges» non può essere specificato poiché stiamo mostrando tutti gli utenti
121	Convalida del documento non riuscita
122	Errore di aggiornamento
124	Errore nell'elaborazione dell'id dello schema
125	Il comando non è riuscito, spesso a causa di un riferimento ambiguo o di una sessione mancante.
126	Il database esiste già
127	Il nome del database esiste già nel nome di un caso diverso
128	Troppi campi
129	In attesa della cancellazione del database
130	Valore del parametro errato
136	La posizione del cursore è stata persa o una funzionalità di change stream richiesta non è abilitata.
139	Errore dell'interprete JS
145	Alias non definito nella query
147	non padrone
148	Risorse insufficienti per eseguire l'interrogazione
149	Alias già definito nella query
154	Un argomento ha un valore illegale o un tipo errato (non numerico).
155	Valore NULL rilevato

Codice di errore	Description
156	Valore NON DEFINITO rilevato
157	Valore NAN rilevato
158	Raccolta già utilizzata per un altro scopo
159	Un valore di durata è superato o un'operazione di rete non è riuscita.
161	La versione di confronto richiesta non è valida o non è disponibile.
163	Un campo o un nome non è valido per l'archiviazione.
164	Un operatore di interrogazione è stato annidato in un modo non consentito.
165	Una vista supera la profondità massima di annidamento di 20 o fa riferimento a se stessa in un ciclo.
166	L'operazione non è supportata su una vista o sono stati utilizzati troppi elementi posizionali.
167	L'operazione o la fase richiesta non è supportata in una visualizzazione.
168	Un'espressione o un operatore di pipeline non è riconosciuto o ha un argomento non valido.
169	BadValue $maxTimeMS ha un valore non integrale
170	BadValue $maxTimeMS non è compreso nell'intervallo
171	Un indice non può essere creato su più campi di matrice.
173	Il valore Skip deve essere positivo
174	La verbosità del log dovrebbe essere un numero
181	Lo schema di tasti corrisponde a più indici; specifica invece l'indice per nome.
197	La specifica dell'indice contiene un campo non valido o non supportato.

Codice di errore	Description
207	uuid deve essere un campo binario a 16 byte con sottotipo UUID (4)
224	Questa funzionalità di interrogazione non è consentita nel contesto corrente.
225	È stata avviata una transazione più recente nella sessione, quindi quella precedente non può procedere.
241	Non è stato possibile convertire un valore nel tipo richiesto in $convert.
250	Visualizzazione non valida
251	Il numero di transazione specificato non corrisponde a nessuna transazione attiva.
252	Limite di ricorsione raggiunto
253	Impossibile inserire la voce nella vista
254	Impossibile eliminare la voce dalla visualizzazione
256	Documento filtrato in base alla clausola $having
257	La dimensione totale di tutte le operazioni della transazione supera il limite consentito.
263	Questa operazione non può essere eseguita all'interno di una transazione con più documenti.
291	impossibile trovare l'indice per la query $GeoNear
300	Autenticazione non riuscita a causa di un handshake SASL o di un errore di sessione.
301	Un meccanismo di autenticazione o un'operazione richiesta non sono supportati.
302	L'autenticazione richiede dati aggiuntivi per completare lo scambio SASL.
303	La configurazione della fase, dell'opzione o del ruolo richiesta non è supportata.

Codice di errore	Description
304	La fase di aggregazione richiesta non è supportata.
305	L'operatore di proiezione di aggregazione non è supportato.
306	L'operatore accumulatore del gruppo di aggregazione specificato non è supportato.
502	La colonna esiste già
505	Inserimento non consentito qui
601	Le coordinate sono fuori limite
602	Errore di indice
700	Le modifiche a questo parametro non sono consentite in modalità live
701	Valore non valido per questo parametro
703	Transizione di stato illegale per questo parametro
999	ID transazione errato
4570	Parametro $lookup che prevede un valore di stringa
4572	In $lookup mancano i campi obbligatori
10065	A un parametro che deve essere un oggetto (come vars o $rand) è stato assegnato un valore non oggetto.
10107	non padrone
11000	Una chiave duplicata viola un vincolo di indice univoco.
11601	L'operazione è stata interrotta.
11603	Sintassi della richiesta del cliente danneggiata o illegale
13103	troppe chiavi composte

Codice di errore	Description
13297	Esiste già un database con lo stesso nome ma differisce solo in lettere maiuscole .
13388	Nome della fase della pipeline non riconosciuto: $collStats
13435	non master e slaveOK=false
14031	Spazio su disco insufficiente
15888	deve indicare il nome della collezione per creare
15947	i campi di un gruppo devono essere specificati in un oggetto
15950	Il nome del campo aggregato del gruppo non può essere un operatore
15952	operatore $group sconosciuto
15955	Errore di sintassi nell'operatore di aggregazione di gruppo: il campo _id deve essere specificato
15956	L'argomento $skip è negativo; deve essere un numero non negativo.
15957	il limite deve essere specificato come numero
15958	Il valore $limit deve essere un numero positivo.
15959	il filtro di corrispondenza deve essere un'espressione in un oggetto
15969	Errore di sintassi: il valore dell'operatore $project deve essere un documento.
15972	L'argomento di $skip deve essere un numero
15973	la specifica della chiave $sort deve essere un oggetto
15974	L'ordine delle chiavi $sort deve essere specificato utilizzando un numero
15975	L'ordine della chiave $sort deve essere 1 (in ordine crescente) o -1 (in ordine decrescente)

Codice di errore	Description
15976	$sort stage deve avere almeno una chiave di ordinamento
15981	il campo di interrogazione non è una stringa
15983	Un oggetto che rappresenta un'espressione deve avere esattamente un campo.
15992	tipo di campo non consentito NULL nell'espressione dell'oggetto
15998	I nomi dei percorsi dei campi non possono essere stringhe vuote.
16006	Un valore non può essere convertito in una data perché non è un tipo di data valido.
16007	Un valore non può essere convertito in una stringa perché era prevista una stringa.
16020	È stato chiamato un operatore di espressione con un numero errato di argomenti.
16034	L'indice iniziale $SubstrBytes deve essere di tipo numerico.
16035	L'argomento relativo alla lunghezza di $SubstrBytes deve essere di tipo numerico.
16395	$where non è consentito all'interno di un'espressione di aggregazione $match
16400	impossibile aggiungere un'espressione per un campo perché esiste già un'espres sione per quel campo o per uno dei suoi sottocampi.
16401	impossibile aggiungere un'espressione per un sottocampo perché esiste già un'espressione che si applica all'intero campo
16404	Le espressioni $ non sono consentite in questo contesto
16406	Non è possibile escludere campi diversi da _id
16410	Il nome o il percorso di un campo $project inizia illegalmente con o contiene '$'.
16411	I nomi dei campi non possono contenere caratteri nulli

Codice di errore	Description
16419	il percorso del campo non deve contenere caratteri nulli incorporati
16424	$near non è consentito all'interno di un'espressione di aggregazione $match
16426	$nearSphere/$NEAR non è consentito all'interno di un'espressione di aggregazi one $match
16435	Il componente della pipeline deve contenere un elemento
16493	Ho provato a creare una stringa più lunga di 16 MB
16554	$add supporta solo tipi numerici o di data.
16605	$geoEar richiede un campo 'near' valido e opzioni GeoNear digitate correttam ente.
16606	$geoNear richiede un'opzione 'distanceField' come stringa
16607	$geoNear richiede che l'opzione data sia specificata come stringa.
16608	Dividi per zero
16610	A $mod è stato dato un divisore pari a zero.
16611	$mod supporta solo i tipi numerici per entrambi gli operandi.
16612	è consentita una sola data in un'espressione $add
16613	L'operazione ha tentato di sottrarre un valore non numerico e non datato da una data.
16702	$concat supporta solo argomenti di tipo stringa.
16730	il testo IndexVersion non è compreso nell'intervallo
16755	non è possibile indicizzare la geometria con un ordine di avvolgimento rigoroso
16800	non può avere 2 campi geografici

Codice di errore	Description
16801	2d deve essere il primo nell'indice
16804	oggetto di posizione previsto, matrice di posizione non nel formato corretto
16810	$mod richiede un array di esattamente due valori come parametro.
16837	Un percorso di aggiornamento non può modificare lo stesso campo più di una volta.
16868	Nome di variabile non valido
16869	Nome di variabile non valido
16870	Nome di variabile non valido
16872	Un percorso di campo è vuoto oppure '$' viene utilizzato solo come percorso di campo in $project.
16873	Fieldpath non inizia con $
16874	$let supporta solo un oggetto come argomento
16875	A $let è stato assegnato un parametro non riconosciuto.
16876	Parametro 'vars' mancante in $let
16877	Parametro 'in' mancante in $let
16878	$map supporta solo un oggetto come argomento
16879	A $map viene assegnato un parametro non riconosciuto.
16880	A $map manca il parametro 'input' richiesto.
16881	Campo obbligatorio mancante
16882	Parametro «in» mancante in $map
16883	L'input per $map deve essere un array.

Codice di errore	Description
16898	$minDistance non deve essere negativo
16899	$maxDistance deve essere un numero
16900	$maxDistance non deve essere negativo
16945	È stato superato il limite di memoria per $group o $lookup. Il pass consente di aderireDiskUse:true .
16949	Opzione di aggregazione non valida; allow DiskUse deve essere un valore booleano.
16990	$out richiede una stringa o un argomento oggetto che nomini la destinazione di output.
16996	inserimento per $out non riuscito
16997	$out non è riuscito perché gli indici della raccolta di destinazione sono cambiati durante l'elaborazione.
17040	L'argomento ElementsTrue di $all deve essere un array.
17041	L'argomento ElementTrue di $any deve essere un array.
17042	Entrambi gli operandi di $set IsSubset devono essere matrici.
17043	Tutti gli operandi di $setUnion devono essere matrici.
17044	Un operando di $setEquals non è un array.
17045	$setEquals richiede almeno due argomenti.
17046	Entrambi gli operandi di $set IsSubset devono essere matrici.
17047	Tutti gli operandi di $setIntersection devono essere matrici.
17048	Entrambi gli operandi di $setDifference devono essere matrici.

Codice di errore	Description
17049	Entrambi gli operandi di $setDifference devono essere matrici.
17053	L'espressione di $redact non dovrebbe restituire nulla a parte le variabili $$KEEP, $$DESCEND e $$PRUNE
17080	Parametro 'if' mancante in cond
17081	Manca il parametro 'then' in cond
17082	Manca il parametro 'else' in cond
17083	Parametro non riconosciuto per $cond
17124	L'argomento di $size deve essere un array.
17275	Impossibile ridefinire ROOT
17276	Variabile non trovata
17280	Il valore della chiave dell'indice è troppo grande per essere indicizzato.
17287	$merge non può modificare la stessa riga di destinazione più di una volta.
17293	l'opzione di indice di testo 'textIndexVersion' deve essere un numero
17302	Il valore di questo argomento deve essere maggiore o uguale a zero.
17304	$geoEar richiede un punto «vicino» e argomenti di geo-query validi.
17308	A $meta viene fornito un argomento non supportato.
17310	Entrambi gli operandi di $set IsSubset devono essere matrici.
17313	$match with $text è consentito solo come prima fase della pipeline.
17320	CreateCollection non può essere eseguito su uno spazio dei nomi contenente un nome riservato.
17385	Impossibile destinare $alla collezione speciale

Codice di errore	Description
17390	$group non supporta le espressioni in stile inclusione
17419	Il documento risultante dopo l'aggiornamento supera la dimensione massima consentita.
17420	Il documento da modificare è più grande di 16777216
18533	Il campo di formato deve essere una stringa
18534	Campo non riconosciuto relativo a $date ToString
18535	La stringa di formato della data non è valida a causa di un segno di percentuale finale.
18536	Formato ToString $data/ora non valido
18537	$date ToString non può formattare un componente di data al di fuori dell'intervallo 0-9999.
18627	Parametro 'format' mancante in $date ToString
18628	Parametro «date» mancante in $date ToString
18629	tipo di argomento errato
25278	$geoNear richiede un argomento 'distanceField'
26823	Un ID di operazione non può essere rappresentato all'interno dei bit disponibili.
28539	Il primo argomento del comando distinct deve essere una stringa
28646	$Filter prevede un oggetto come parametro
28647	Campo imprevisto per $filter
28648	$Filter richiede un campo 'input'
28649	$Filter richiede un campo «as»

Codice di errore	Description
28650	$Filter richiede un campo 'cond'
28651	È richiesto l'input dell'array
28656	$substrBytes: intervallo non valido, l'indice iniziale è un byte di continuazione. UTF-8
28657	$substrBytes: intervallo non valido, l'indice finale è al centro di un carattere. UTF-8
28664	$concatArrays supporta solo gli input di array.
28667	Un'espressione come $slice, $range o $index OfArray ha ottenuto un numero errato di argomenti.
28680	$abs viene applicato a un valore, ad esempio il long minimo, che non può essere negato.
28689	L'argomento di un operatore deve essere un array ma è stato fornito un tipo diverso.
28690	Il secondo argomento dell'array ElemAt deve essere numerico
28691	Il secondo argomento dell'array ElemAt deve essere rappresentato da un numero intero a 32 bit
28714	L'argomento di $sqrt deve essere maggiore o uguale a 0
28724	Il primo argomento di $slice deve essere un array.
28725	Il secondo argomento di $slice deve essere un valore numerico.
28726	Il secondo argomento di $slice deve contenere un numero intero a 32 bit.
28727	Il terzo argomento per $slice deve essere un numero.
28728	Il terzo argomento di $slice deve essere un numero intero a 32 bit.

Codice di errore	Description
28729	Il terzo argomento di $slice deve essere un numero intero positivo a 32 bit.
28745	la specifica $sample stage deve essere un oggetto
28746	l'argomento relativo alla dimensione di $sample deve essere un numero
28747	l'argomento relativo alla dimensione di $sample non deve essere negativo
28748	$sample ha ricevuto un'opzione non riconosciuta.
28749	$sample stage deve specificare una dimensione
28756	L'argomento di $log deve essere numerico.
28757	$log richiede che il suo argomento di base sia un valore numerico.
28758	L'argomento di $log deve essere un numero positivo.
28759	La base di $log deve essere un numero positivo diverso da 1.
28761	L'argomento di $log10 deve essere un numero positivo.
28762	La base di $pow deve essere un valore numerico.
28763	L'esponente di $pow deve essere numerico.
28764	$pow non può assumere una base pari a 0 e un esponente negativo
28765	$ceil, $exp, $floor, $ln o $log10 riceve un tipo non numerico.
28766	$ln richiede un argomento numerico positivo.
28812	Fieldpath è vuoto
28818	Fieldpath non inizia con '$'
31022	Questo operatore richiede la specificazione di un parametro «input».
31023	Questo operatore richiede un parametro 'regex'.

Codice di errore	Description
31024	All'operatore viene fornito un argomento sconosciuto.
31032	Il nome di un campo non può contenere un byte nullo incorporato.
31034	Un valore non rientra nell'intervallo consentito di [-32768, 32767].
31095	WeekYear'iso' deve restituire un numero intero compreso tra 1 e 9999.
31138	Il valore $meta utilizzato per l'ordinamento non è una parola chiave di metadati valida.
31170	Era previsto un oggetto vuoto ma è stato fornito un valore non vuoto.
31249	Un percorso di proiezione si scontra con un altro campo della proiezione.
31254	Non è possibile specificare un'inclusione all'interno di una proiezione di esclusion e.
31393	$BSonSize richiede un documento come input.
31394	Un operatore di proiezione posizionale non può apparire al centro di un percorso di campo.
34413	Un operatore non valido è stato utilizzato in una query geo near.
34435	L'argomento di $reverseArray deve essere un array.
34443	$range richiede un valore iniziale numerico.
34444	$range richiede un valore iniziale rappresentabile come numero intero a 32 bit.
34445	$range richiede che il suo valore finale sia numerico.
34446	Il valore finale di $range deve rientrare in un numero intero a 32 bit.
34447	$range richiede un valore numerico a gradini.
34448	$range richiede un valore graduale rappresentabile come numero intero a 32 bit.

Codice di errore	Description
34449	$range richiede un valore di step diverso da zero
34450	L'indice iniziale di $substrCP deve essere di tipo numerico.
34451	L'indice iniziale $SubstrCP non può essere rappresentato come un numero intero a 32 bit.
34452	L'argomento relativo alla lunghezza di $SubstrCP non è un tipo numerico.
34453	La lunghezza di $SubstrCP non può essere rappresentata come un numero intero a 32 bit.
34454	$substrCP: la lunghezza deve essere un numero intero non negativo.
34455	$substrCP: l'indice iniziale deve essere un numero intero non negativo.
34460	$zip supporta solo un oggetto come argomento.
34461	Gli input $zip devono essere un array di espressioni.
34462	L'opzione 'defaults' deve essere un array di espressioni.
34463	L'LongestLength opzione use di $zip deve essere booleana.
34464	A $zip è stato fornito un argomento sconosciuto.
34465	$zip richiede almeno un array di input
34466	non può specificare i valori predefiniti a meno che l'uso non sia vero LongestLe ngth
34467	i valori predefiniti e gli ingressi devono avere la stessa lunghezza
34468	$zip riceve un'espressione non di matrice nel suo input.
34471	$strLenCP richiede un argomento di tipo stringa.
34473	$str LenBytes richiede un argomento di tipo stringa.

Codice di errore	Description
40060	$switch richiede un oggetto come argomento.
40061	$switch richiede che 'branch' sia un array.
40062	$switch richiede che ogni ramo sia un oggetto.
40063	A $switch viene assegnato un argomento sconosciuto all'interno di un ramo.
40064	$switch richiede che ogni ramo abbia un'espressione 'case '.
40065	$switch richiede che ogni ramo abbia un'espressione 'then '.
40066	$switch non è riuscito a trovare un ramo corrispondente per un input e non è stato specificato alcun valore predefinito.
40067	L'espressione $switch contiene un argomento sconosciuto.
40068	$switch richiede almeno una filiale
40075	$reduce richiede un oggetto
40076	Argomento sconosciuto per $reduce
40077	$reduce richiede un 'input'
40078	$reduce richiede 'initialValue'
40079	$reduce richiede «in»
40080	L'input per $reduce deve essere un array.
40081	L'operatore richiede un array come secondo argomento.
40085	$split richiede il suo primo argomento per restituire una stringa.
40086	Il secondo argomento di $split deve restituire una stringa.
40087	$split richiede un separatore non vuoto

Codice di errore	Description
40090	$index OfArray richiede un array come primo argomento
40091	Il primo argomento di $index non OfBytes è una stringa.
40092	$index OfBytes richiede una stringa come secondo argomento.
40093	$indexOfCP richiede una stringa come primo argomento.
40094	$indexOfCP richiede una stringa come secondo argomento.
40096	$index OfArray richiede valori di indice iniziali e finali integrali.
40097	$index OfArray richiede indici numerici di inizio e fine non negativi.
40148	il ByCount campo di ordinamento deve essere definito come un percorso con prefisso $ o un'espressione all'interno di un oggetto
40149	il ByCount campo di ordinamento deve essere specificato come stringa o come oggetto
40156	il campo di conteggio deve essere una stringa non vuota
40157	il campo di conteggio deve essere una stringa non vuota
40158	il campo count non può essere un percorso con prefisso $
40159	il campo count non può contenere un byte nullo
40160	il campo di conteggio non può contenere '.'
40169	la specifica $facet deve essere un oggetto non vuoto
40170	gli argomenti di $facet devono essere matrici
40171	gli elementi degli array in $facet spec devono essere oggetti non vuoti
40172	la sottopipeline in $facet stage non può essere vuota
40176	Nomi di campo in conflitto forniti

Codice di errore	Description
40177	$project e $addFields richiedono almeno un campo di output.
40178	Non è possibile escludere campi tranne '_id' in una proiezione di inclusione
40179	Non è possibile includere campi o reimpostare campi o aggiungere campi calcolati in una proiezione di esclusione
40180	$project non consente un oggetto vuoto come valore di campo.
40181	$project:: non valido causato da:: una specifica di espressione deve contenere esattamente un campo.
40183	Un nome di campo punteggiato non può essere utilizzato all'interno di un sottooggetto.
40192	Il campo $bucket 'bounders' deve avere almeno 2 valori
40193	Tutti i valori nell'opzione 'bounders' di $bucket devono avere lo stesso tipo
40194	L'opzione 'bounders' di $bucket deve essere ordinata, ma gli elementi non sono in ordine crescente
40195	Il campo 'default' $bucket deve essere un'espressione costante
40196	Il campo 'output' $bucket deve essere un oggetto
40197	Opzione non riconosciuta per $bucket
40198	$bucket richiede la specificazione di 'groupBy' e 'bounders'
40199	Il campo 'default' $bucket deve essere inferiore al limite più basso o maggiore o uguale al limite più alto.
40200	Il campo $bucket 'bounders' deve essere un array
40201	L'argomento di $bucket stage deve essere un oggetto

Codice di errore	Description
40202	Il campo 'groupBy' $bucket deve essere definito come un percorso o un'espressione con prefisso $
40203	Il campo deve essere un oggetto accumulatore
40204	Il nome del campo non può contenere «.»
40205	Il nome del campo non può essere il nome di un operatore
40206	L'accumulatore è un operatore unario
40218	la query richiede i metadati del punteggio di testo, ma non è disponibile
40228	L'espressione 'NewRoot' $replaceRoot deve restituire un oggetto.
40229	$replaceRoot richiede un oggetto come specifica.
40230	$replaceRoot accetta solo l'opzione 'NewRoot '.
40234	Il campo $group deve essere un oggetto accumulatore.
40235	Il nome di un campo non può contenere un carattere «.».
40236	I nomi dei campi non possono iniziare con '$'
40237	Questo accumulatore è un operatore unario e accetta un solo argomento.
40239	Il campo 'groupBy' $bucketAuto deve essere definito come un percorso con prefisso $ o un oggetto espressione
40240	L'argomento di $BucketAuto deve essere un oggetto
40241	Il campo 'buckets' $bucketAuto deve essere un valore numerico
40242	Il campo 'buckets' $bucketAuto deve essere rappresentabile come un numero intero a 32 bit
40243	Il campo 'buckets' $bucketAuto deve essere maggiore di 0

Codice di errore	Description
40244	Il campo 'output' $bucketAuto deve essere un oggetto
40245	Opzione non riconosciuta per $bucketAuto
40246	$bucketAuto richiede la specificazione di 'groupBy' e 'buckets'
40257	Granularità di arrotondamento sconosciuta
40258	$bucketAuto può specificare una «granularità» con solo limiti numerici
40260	$BucketAuto può specificare una «granularità» solo con numeri non negativi, ma ha trovato un numero negativo
40261	Il campo 'granularity' $bucketAuto deve essere una stringa
40272	La specifica della fase $addFields deve essere un oggetto.
40319	la specifica $lookup stage deve essere un oggetto
40320	manca l'opzione 'from' nella specifica della fase $lookup
40321	l'opzione 'from' di $lookup deve essere una stringa
40322	namespace $lookup non valido
40323	Un oggetto di specifica dello stadio della pipeline deve contenere esattamente un campo.
40324	La pipeline contiene un nome d'arte non riconosciuto.
40331	Questa fase di aggregazione non è consentita all'interno di una pipeline $facet.
40352	I percorsi dei campi $project non possono essere stringhe vuote.
40353	Il percorso di un campo non deve terminare con un '.'.
40386	$array ToObject richiede un input di matrice.
40390	$object ToArray richiede un documento come input.

Codice di errore	Description
40391	$array ToObject richiede un formato di input coerente. Gli elementi devono essere tutti matrici o essere tutti oggetti. L'oggetto è stato rilevato, ora è stato trovato: array
40392	$array ToObject richiede oggetti con esattamente le chiavi 'k' e 'v'.
40393	$array ToObject richiede un oggetto con le chiavi 'k' e 'v'. Mancano una o entrambe le chiavi
40394	$array ToObject richiede oggetti con chiavi di tipo stringa 'k' e valori 'v'.
40395	$array ToObject richiede coppie chiave-valore le cui chiavi sono stringhe.
40396	$array ToObject richiede un formato di input coerente. Gli elementi devono essere tutti matrici o essere tutti oggetti. L'array è stato rilevato, ora è stato trovato: oggetto
40397	$array ToObject richiede un array di matrici a due elementi.
40398	A $array ToObject viene assegnato un input di tipo o formato non riconosciuto.
40400	$mergeObjects richiede che tutti gli input siano oggetti.
40414	Un campo di comando obbligatorio è mancante o la convalida non è riuscita.
40415	Il comando contiene un campo non riconosciuto.
40422	Il nome del campo di un array BSON ha un valore non valido.
40485	È stato fornito un identificatore di fuso orario non riconosciuto.
40489	$date FromParts non consente di mescolare i normali campi del calendario con i campi ISO relativi alla data settimanale
40515	Un FromParts argomento $date non restituisce un numero intero.
40516	$date FromParts richiede la presenza di «year» o «isoWeekYear»

Codice di errore	Description
40517	L'opzione 'timezone' deve restituire una stringa per gli operatori di data.
40518	$date FromParts o $date ToParts hanno un parametro non riconosciuto o digitato erroneamente.
40519	$date supporta FromParts solo un oggetto come argomento.
40520	A $date ToParts è stato assegnato un parametro non riconosciuto.
40522	$date ToParts richiede un argomento oggetto con un parametro 'data'.
40523	$date FromParts 'year' deve essere un numero intero compreso tra 1 e 9999.
40525	$date week/day I valori FromParts ISO non sono compresi nell'intervallo o mancano i campi obbligatori.
40533	L'argomento del fuso orario deve essere una stringa.
40535	Allo stage o all'espressione è stata fornita un'opzione non riconosciuta.
40536	L'espressione accetta esattamente un argomento quando viene fornita una matrice.
40539	Manca l'argomento «data» richiesto per un operatore di data.
40540	$date supporta FromString solo un oggetto come argomento
40541	$date FromString ha ricevuto un argomento non riconosciuto.
40542	Parametro 'dateString' mancante in $date FromString
40543	$date FromString richiede che 'DateString' sia una stringa.
40545	La date/time stringa è incompleta, mancano gli elementi obbligatori.
40551	non è possibile passare una date/time stringa con informazioni sul fuso orario insieme a un argomento sul fuso orario

Codice di errore	Description
40553	$date FromString non può analizzare la stringa di data specificata.
40554	non è possibile passare una date/time stringa con offset GMT insieme a un argomento relativo al fuso orario
40575	$changeStream 'FullDocument' deve essere «predefinito» o «updateLookup».
40600	$out non può essere utilizzato all'interno di una fase $facet.
40601	$out e $merge possono essere solo la fase finale della pipeline.
40602	Una fase come $ChangeStream o $GeoNear non è la prima in cantiere.
40603	$GeoNear non è stata la prima fase della pipeline dopo l'ottimizzazione. L'ottimiz zazione è disabilitata o inibita?
40647	Il token di curriculum del change stream ha un campo _data mancante o digitato erroneamente.
40674	È consentito un solo tipo di opzione di curriculum, ma ne sono state trovate diverse.
50001	auto non IndexId:false è consentito per la raccolta specificata.
50694	A questo operatore è stata fornita un'argomentazione sconosciuta.
50695	Questo operatore richiede un campo «input».
50696	L'operatore supporta solo un oggetto come argomento.
50699	Questo operatore di espressione richiede che il suo input sia una stringa.
50700	L'argomento 'chars' deve essere una stringa.
50736	GetMore non può essere eseguito in una sessione su un cursore non creato in una sessione.

Codice di errore	Description
50737	L'esecuzione di GetMore su un cursore creato dalla sessione richiede la fornitura di un lsid.
50752	L'indice iniziale di $substrBytes è negativo.
50808	La fase $changeStream prevede un documento come argomento.
51024	Il valore di 'forza' delle regole di confronto deve essere pari o inferiore a 5.
51047	Questo operatore o fase non è consentito all'interno di una sottopipeline $lookup.
51075	Impossibile impostare le opzioni sia in $regex che in $options
51076	Memoria insufficiente per creare l'indice vettoriale per il numero specificato di elenchi.
51078	Impossibile trovare il nome dell'indice.
51079	Le statistiche di latenza non sono supportate
51091	Un'espressione regolare non è valida o è mal formata.
51103	L'operatore si aspetta un oggetto con argomenti denominati ma ne ha ricevuto un altro tipo.
51104	L'operatore richiede che il suo argomento 'input' sia una stringa.
51105	L'argomento 'regex' di un operatore non è una stringa o un tipo regex.
51106	Un operatore richiede che il suo parametro 'options' sia una stringa.
51107	Le opzioni $regex non possono essere specificate in entrambi i campi 'regex' e 'options'.
51108	Le opzioni di espressione regolare contengono un flag non valido o un byte nullo incorporato.
51109	Un'espressione regolare non può contenere un byte nullo incorporato.

Codice di errore	Description
51110	Le opzioni di espressione regolare non possono contenere un byte nullo incorporato.
51111	L'espressione regolare fornita a questo operatore non è valida.
51151	La dimensione del buffer per memorizzare l'output ha superato il limite del tipo di dati
51156	Impossibile eseguire un'espressione regolare in $regex durante la valutazione.
51270	Un oggetto di sottoproiezione vuoto non è un valore di proiezione valido.
51272	Errore di sintassi: $projection richiede almeno un campo di output.
51744	L'operatore richiede che 'replacement' sia una stringa.
51745	L'operatore richiede che l'argomento 'find' sia una stringa.
51746	L'argomento 'input' di un operatore non è una stringa.
51747	Un operatore richiede un parametro «sostitutivo».
51748	Questo operatore richiede un parametro 'find'.
51749	Questo operatore richiede un parametro «input».
51750	Un operatore di aggregazione ha ricevuto un argomento sconosciuto.
100003	Errore interno di sistema durante l'elaborazione del percorso di interrogazione ottimizzato
100005	È stato superato un limite configurato, ad esempio raccolte, ruoli personalizzati o privilegi.
100006	È stato superato il numero massimo di utenti consentiti.
3040501	$rand attualmente non accetta argomenti

Codice di errore	Description
3041701	$getField ha ricevuto un argomento sconosciuto in $project.
3041702	$project:: non valido causato da:: $getField richiede l'indicazione di un 'campo'
3041703	$project:: non valido causato da:: $getField richiede la specificazione di 'input'
3041704	$getField richiede l'argomento 'field' per restituire una stringa.
4161100	$setField e $unsetField supportano solo un oggetto come argomento.
4161101	A $setField o $unsetField è stato assegnato un argomento sconosciuto.
4161102	$setField e $unsetField richiedono la specificazione di un 'campo'.
4161103	$project:: non valido causato da:: $setField richiede l'indicazione di un 'valore'
4161105	$setField o $unsetField richiedono un 'input' per la valutazione di un oggetto.
4161106	$setField e $unsetField richiedono un 'campo' per restituire una costante.
4161107	$setField o $unsetField richiede un 'campo' per restituire una stringa.
4161108	$project non consente un riferimento al percorso di campo qui; usa invece {$literal:...}.
4161109	$setField e $unsetField richiedono la specificazione di un campo 'input'.
4662500	Il comando contiene un campo non riconosciuto.
4862100	Lo spazio dei nomi completo supera la lunghezza massima consentita.
5166301	$DateDiff supporta solo un oggetto come argomento
5166302	A $DateDiff è stato assegnato un argomento non riconosciuto.
5166303	Parametro 'startDate' mancante in $DateDiff
5166304	Parametro 'endDate' mancante in $DateDiff

Codice di errore	Description
5166305	Parametro 'unit' mancante in $dateDiff
5166307	Un argomento $dateDiff non è una data.
5166308	$DateDiff è andato in overflow durante il calcolo della differenza di data.
5166400	$dateAdd e $dateSubtract richiedono che il loro argomento sia un oggetto.
5166401	$dateAdd o $dateSubtract hanno ottenuto un argomento non riconosciuto.
5166402	$dateAdd o $dateSubtract richiedono la presenza di startDate, unit e amount.
5166403	$dateAdd richiede che startDate sia convertibile in una data
5166404	$dateAdd prevede una stringa che definisca l'unità di tempo
5166405	$dateAdd prevede un numero intero di unità di tempo
5166406	Data Add overfloccata
5166503	È stato fornito un valore di unità di tempo sconosciuto.
5439002	$DateTrunc è stato sovraccaricato durante il calcolo della data troncata.
5439006	Valore binSize non supportato per DateTrunC
5439007	$DateTrunc supporta solo un oggetto come argomento
5439008	Argomento $DateTrunc non riconosciuto; data prevista, unità, BinSize, fuso orario, inizio. OfWeek
5439009	Parametro «date» mancante in $DateTrunc
5439010	Parametro 'unit' mancante in $DateTrunc
5439012	$dateTrunc richiede che 'date' sia una data.
5439013	$dateDiff e $dateTrunc richiedono che 'unit' sia una stringa.

Codice di errore	Description
5439014	Il valore 'unità' di $dateDiff e $DateTrunc non è un'unità di tempo riconosciuta.
5439015	$dateDiff o $dateTrunc richiedono che 'start' sia una stringa. OfWeek
5439016	Il valore OfWeek 'start' di $dateDiff o $dateTrunc non è un giorno della settimana valido.
5439017	$DateTrunc richiede che 'binSize' sia un numero intero a 64 bit.
5439018	$DateTrunc richiede che 'binSize' sia maggiore di 0.
5447000	Lo stage $collStats deve essere un oggetto annidato.
5626500	$geoNear, $near e $nearSphere non sono consentiti in questo contesto, poiché questi operatori richiedono l'ordinamento dei dati geospaziali. Se non hai bisogno dell'ordinamento, considera invece l'utilizzo di $geoWithin.
5654600	$project non consente un riferimento al percorso di campo in questo contesto.
5654601	$project:: non valido causato da:: $getField richiede 'field' per restituire una costante, ma ha ottenuto un argomento non costante
5654602	$getField richiede un 'field' per restituire una stringa.
5733415	non posso $mod per zero
5860400	$geoNear richiede un argomento «vicino»
5887502	Tutti gli operandi di $setEquals devono essere matrici.
5911200	PlanExecutor errore durante l'aggregazione:: causato da: $mergeObjects supporta solo oggetti
5976500	A $dateAdd è stato assegnato un valore del parametro 'amount' non valido.
6045000	A $dateSubtract è stato assegnato un valore del parametro 'amount' non valido.

Codice di errore	Description
7158303	$switch non ha trovato alcun ramo corrispondente per l'input e non è stato specificato alcun valore predefinito.

Note di rilascio

Queste note di rilascio descrivono le funzionalità, i miglioramenti e le correzioni di bug di Amazon DocumentDB organizzate per versione principale del motore. All'interno di ogni sezione delle versioni, le voci sono elencate per data in un formato tabellare.

Per dettagli sulla numerazione delle versioni del motore e delle versioni delle patch (inclusa la mappatura tra i prefissi delle versioni delle patch del motore e le versioni del motore Amazon DocumentDB), consulta. Numerazione delle versioni del motore

Puoi determinare la versione attuale della patch del motore Amazon DocumentDB eseguendo il seguente comando:

```
db.runCommand({getEngineVersion: 1})
```

> **Note**
>
> A partire da Amazon DocumentDB 5.0.2 e 8.0.2, questo comando restituisce la versione del motore e la versione della patch del motore come una singola stringa nel modulo. *`engine-version`*+*`engine-patch-version`* Ad esempio, restituisce un cluster su Amazon DocumentDB 5.0.2 e un cluster su `5.0.2+3.0.12345` 8.0.2 restituisce. `8.0.2+4.0.12345` I cluster nelle versioni precedenti restituiscono da soli la versione della patch del motore, ad esempio. `3.0.12345` Per ulteriori informazioni, consulta Numerazione delle versioni del motore.

Se sul cluster non è installata la versione più recente del motore, è probabile che sia disponibile una manutenzione in sospeso per l'aggiornamento del motore. Per ulteriori informazioni, consulta Manutenzione di Amazon DocumentDB la Guida per gli sviluppatori.

Argomenti

- Note di rilascio della versione del motore Amazon DocumentDB
- Note di rilascio generali di Amazon DocumentDB
- Note sulla versione dei cluster elastici Amazon DocumentDB

Note di rilascio della versione del motore Amazon DocumentDB

Note di rilascio per le versioni del motore Amazon DocumentDB organizzate per versione principale.

Amazon DocumentDB 8.0

Note di rilascio per la versione 8.0 del motore Amazon DocumentDB (compatibilità con MongoDB 8.0).

Amazon DocumentDB 8.0.2

Data	Note di rilascio
28 settembre 2026	Amazon DocumentDB 8.0.2 (versione del motore: 4.0.12042) Nuove funzionalità e miglioramenti 1. È stato aggiunto il supporto per la fase di `$setWindowFields` aggregazione e i relativi operatori di finestra nella versione 3 di Planner. 2. È stato aggiunto il supporto per la fase di `$bucketAuto` aggregazione nella versione 3 di Planner. 3. È stato aggiunto il supporto per la fase di `$facet` aggregazione nella versione 3 di Planner. 4. È stato aggiunto il supporto per la fase di `$graphLookup` aggregazione nella versione 3 di Planner. 5. È stato aggiunto il supporto per le `$lookup` sottoquery correlate , inclusa la sintassi concisa delle sottoquery correlate, con un sottoinsieme di fasi della pipeline nella versione 3 di Planner. 6. È `$changeStreamSplitLargeEvent` stato aggiunto il supporto per la fase di aggregazione. 7. I flussi di modifica ora emettono eventi per le operazioni. `createCollection` 8. I flussi di modifica ora emettono eventi per le operazioni. `createIndex` 9. Prestazioni di scrittura del flusso di modifica migliorate.

Data	Note di rilascio
	10. È stato aggiunto il supporto per le scritture riutilizzabili. 11. È stato aggiunto il supporto per il `reIndex` comando sugli indici parziali. 12. La versione 3 di Planner ora supporta le scansioni degli indici sui predicati. `$expr` 13. La versione 3 di Planner introduce Index Only Scans (IXONLYSCAN) a più campi per le query di ricerca. Per ulteriori informazioni, consulta Index-only scansioni (domande coperte). 14. La versione 3 di Planner introduce Index Only Scans (IXONLYSCAN) per fasi di aggregazione come. `$group` Per ulteriori informazioni, consulta Index-only scansioni (domande coperte). 15. Ottimizzazione delle prestazioni per `$count` e nella versione 3 di Planner. `countDocuments()` 16. È stato aggiunto il supporto per l'ordinamento incrementale nella versione 3 di Planner. Per ulteriori informazioni, consulta Ordinamento incrementale. 17. Selezione del piano migliorata per le interrogazioni sull'indice di testo. 18. Il piano di spiegazione per coalesced `$lookup` e `$unwind` stages ora riflette il parametro. `includeArrayIndex` 19. È stato aggiunto il supporto per il filtro degli indici in tutti i comandi. 20. Il pianificatore di query ora dà la priorità agli indici univoci per i predicati di quality (). `$eq` 21. È stato aumentato il limite di troncamento per i comandi aggregati in output. `currentOp` Correzioni di bug 1. È stato corretto un bug nella versione 1 (predefinita) del pianificatore di query. 2. È stata corretta la gestione parziale del testo dell'espressione dell'indice durante il Major Version Upgrade (MVU).

Data	Note di rilascio
	3. È stato risolto un problema relativo al profiler per le query di conteggio rapido. 4. Correzione di sicurezza per il comando. `collStats`

Amazon DocumentDB 8.0.1

Data	Note di rilascio
15 settembre 2026	Miglioramenti della sicurezza e della stabilità Amazon DocumentDB 8.0.1 (versione del motore: 4.0.11638) [in corso] Amazon DocumentDB applica automaticamente questa patch durante la prossima finestra di manutenzione.
31 agosto 2026	Nuova funzionalità Amazon DocumentDB ora supporta gli aggiornamenti delle versioni principali (MVU) sul posto direttamente dalle versioni 3.6 e 4.0 del motore alla versione 8.0. Ciò elimina la necessità di aggiornamenti di versione intermedi e consente di aggiornare i cluster 3.6 o 4.0 preservando i dati, le configurazioni e le impostazioni dei cluster esistenti. Gli aggiornamenti delle versioni principali da Amazon DocumentDB 3.6 e 4.0 a 8.0 sono disponibili in tutte le regioni in cui queste versioni sono attualmente supportate. AWS Per ulteriori informazioni, consulta Aggiornamento della versione principale in-place di Amazon DocumentDB.
18 agosto 2026	Miglioramenti della sicurezza e della stabilità Amazon DocumentDB 8.0.1 (versione del motore: 4.0.9938) Questa patch del motore offre i seguenti miglioramenti:

Data	Note di rilascio
	• Una migliore elaborazione delle richieste Amazon DocumentDB ora convalida le richieste dei client in arrivo in modo più accurato lungo il percorso di connessione e analisi dei messaggi. Ciò migliora la resilienza e la stabilità in una gamma più ampia di modelli di input. • Il rafforzamento dell'handshake di autenticazione Amazon DocumentDB ora convalida lo scambio di autenticazione in modo più completo. Ciò migliora la solidità nella gestione di input imprevisti o non validi. Amazon DocumentDB applica automaticamente questa patch durante la prossima finestra di manutenzione.
11 agosto 2026	Nuove funzionalità Amazon DocumentDB 8.0.1 (versione del motore: 4.0.8864) • Supporto abilitato per l'autenticazione. SCRAM-SHA-256 • RPO ottimizzabile abilitato per i cluster globali di Amazon DocumentDB. • Migliore recupero dello storage durante la garbage collection in background per ridurre lo spazio di archiviazione inutilizzato nel tempo.
22 luglio 2026	Nuova funzionalità I cluster Amazon DocumentDB che eseguono Engine versione 8.0 e successive supportano ora un limite di archiviazione massimo di 256 TiB, raddoppiando il limite precedente di 128 TiB. Per ulteriori informazioni, consulta Quote Amazon DocumentDB.

Data	Note di rilascio
13 luglio 2026	Amazon DocumentDB 8.0.1 (versione del motore: 4.0.8864) Nuove funzionalità e miglioramenti 1. Supporto per 13 operatori di accumulazione nel Planner predefini to (v3):`$top`,,,,,,,,,`$bottom`,,`$topN`,`$bottomN`,`$count`,`$firstN`. `$lastN` `$minN` `$maxN` `$median` `$percentile` `$stdDevPop` `$stdDevSamp` 2. Supporto per 36 operatori di espressione nel Planner predefini to (v3): `$median``$percentile` ,`$stdDevPop` ,`$stdDevSa mp` ,`$round`,`$sigmoid`,`$trunc`,`$firstN`,`$lastN`,,`$maxN`,`$minN`,`$bitAn ze` ,`$bsonSize` ,`$tsIncrement` ,`$tsSecond` ,`$cos`,`$acos`,`$cosh`,`$acosh`,`$sin`,,`$asin`,`$sinh`,`$asinh`,`$tan`,`$atan`,`$ $atanh` `$degreesToRadians` `$radiansToDegrees` `$isNumber` `$toUUID` 3. Supporto per `$sampleRate` Planner predefinito (v3). `$match` 4. Supporto per `$sortByCount` Planner predefinito (v3). 5. È stato aggiunto il supporto per la fase `$listSearchIndexes` di aggregazione nel Planner predefinito (v3). Restituisce i metadati dell'indice di ricerca (nome, stato, interrogabile, versione) per gli indici di ricerca di testo e vettoriali su una raccolta. 6. Supporta la compressione rapida utilizzando il tipo di messaggio OP_COMPRESSED. 7. Supporta i metodi `min` e `max` cursor sul comando find nel Planner predefinito (v3). 8. La versione 3 di Planner introduce Index Only Scans (IXONLYSC AN) per. `countDocuments()` 9. È stato aggiunto il supporto per gli indici nascosti per tutti i tipi di indice (Planner versione 2 e successive). Correzioni di bug

Data	Note di rilascio
	1. È stato risolto un problema per cui le query di `$text` ricerca contenenti caratteri di sottolineatura potevano restituire risultati errati se valutate come filtro residuo.

Amazon DocumentDB 8.0.0

Data	Note di rilascio
15 settembre 2026	Miglioramenti della sicurezza e della stabilità Amazon DocumentDB 8.0.0 (versione del motore: 4.0.11440) [in corso] Correzioni di bug Questa versione include la seguente correzione di bug: 1. È stato risolto un problema per cui le query di `$text` ricerca contenenti caratteri di sottolineatura potevano restituire risultati errati se valutate come filtro residuo. Amazon DocumentDB applica automaticamente questa patch durante la prossima finestra di manutenzione.
18 agosto 2026	Miglioramenti della sicurezza e della stabilità Amazon DocumentDB 8.0.0 (versione del motore: 4.0.9912) Questa patch del motore offre i seguenti miglioramenti: • Una migliore elaborazione delle richieste Amazon DocumentDB ora convalida le richieste dei client in arrivo in modo più accurato lungo il percorso di connessione e analisi dei messaggi. Ciò migliora la resilienza e la stabilità in una gamma più ampia di modelli di input. • Il rafforzamento dell'handshake di autenticazione Amazon DocumentDB ora convalida lo scambio di autenticazione in modo

Data	Note di rilascio
	più completo. Ciò migliora la solidità nella gestione di input imprevisti o non validi. Se non la applichi prima, Amazon DocumentDB applica automaticamente questa patch durante la prossima finestra di manutenzione, 30 giorni dopo il rilascio.
22 luglio 2026	Nuova funzionalità I cluster Amazon DocumentDB che eseguono Engine versione 8.0 e successive supportano ora un limite di archiviazione massimo di 256 TiB, raddoppiando il limite precedente di 128 TiB. Per ulteriori informazioni, consulta Quote Amazon DocumentDB.
13 maggio 2026	Nuova funzionalità Amazon DocumentDB serverless è ora disponibile per Amazon DocumentDB 8.0. Per ulteriori informazioni, consulta Utilizzo di Amazon DocumentDB serverless.
6 maggio 2026	Correzioni di bug e altre modifiche Amazon DocumentDB 8.0 (versione del motore: 4.0.6940) Questo aggiornamento fornisce una patch del motore opzionale con i seguenti miglioramenti: 1. È stato corretto un bug nella Garbage Collection che, in caso limite, poteva causare l'arresto anomalo dei lettori.
20 aprile 2026	Nuova funzionalità In-place L'aggiornamento della versione principale (MVU) da Amazon DocumentDB 5.0 ad Amazon DocumentDB 8.0 è ora disponibile. Per ulteriori informazioni, consulta Aggiornamento della versione principale in-place di Amazon DocumentDB.

Data	Note di rilascio
20 aprile 2026	Nuove funzionalità e miglioramenti Amazon DocumentDB 8.0 (versioni del motore: 4.0.5995, 4.0.6473) Questa è una patch del motore obbligatoria che include le seguenti nuove funzionalità, miglioramenti e correzioni: 1. `$sortArray` operatore in`$project`: • Supporto per `$sortArray` l'operatore nella `$project` fase della pipeline di aggregazione per Planner versione 3. L'`$sortArray` operatore ordina un array in base ai suoi elementi (per i valori scalari) o su un campo specificato (per gli array di documenti), restituendo l'array ordinato in ordine crescente () o decrescente (). `1` `-1` 2. Supporto per,, utilizzando Planner `$out` v3`$collStats` . `$indexStats` `$geoNear` Con Planner V3, lo `$out` stage supporta anche la scrittura su raccolte di diversi database. 3. La pipeline di aggregazione ora si fonde `$lookup` e si organizza anche quando la `$unwind` fase include il parametro. `$unwind` `includeArrayIndex` In precedenza, questa ottimizzazione veniva ignorata quando era presente. `includeArrayIndex` 4. Miglioramenti delle prestazioni per migliorare il throughput di scrittura per db.r8g.8xlarge e le istanze più grandi eseguite in configurazione. I/O-Optimized 5. Ora puoi `Connections (active)` monitorare la metrica per le istanze. Per ulteriori informazioni, consulta Quote delle istanze. 6. Supporto `$dateToParts` e operatori di `$dateFromParts` aggregazione. 7. Command-level le metriche delle prestazioni sono ora disponibili CloudWatch per le operazioni CRUD (`find`,`insert`,`findAndModify` ,,`update`,`delete`,`aggregate` ,`count`,`distinct`,`commitTransaction`) `getMoreabortTransaction` , l'esposizione `AvgDuration` e

Data	Note di rilascio
	`MaxConcurrent` le dimensioni per un `P100Duration` migliore monitoraggio operativo. 8. È stato corretto un bug nel `$replaceRoot` comportamento per cui nel documento di output 1 mancavano `newRoot` campi contenenti valori booleani di`true`/`false`o valori numerici di`0`/. 9. È stato corretto un bug in `$indexOfArray` cui le query valide restituivano errori che indicavano che non erano valide. 10. Il piano di spiegazione ora include le `docsExamined` metriche nell'output. `executionStats` 11. Prestazioni di autenticazione SASL migliorate. 12. Aggiornamenti ottimizzati dei documenti per ridurre il sovraccarico di archiviazione quando i valori dell'indice rimangono invariati e i documenti si adattano alle pagine esistenti. 13. Supporto per `bypassDocumentValidation` in. `$merge` Tutti i cluster Amazon DocumentDB 8.0 verranno aggiornati automatic amente a questa versione di patch durante la prossima finestra di manutenzione.
26 marzo 2026	Nuove funzionalità Amazon DocumentDB ora supporta l'ultima generazione di istanze ottimizzate per la memoria, db.r8g, alimentate da processori Arm-based AWS Graviton4, che forniscono prestazioni fino al 30% migliori rispetto alle istanze R6G. Per ulteriori Gestione delle classi di istanze informazioni, consulta questa guida

Data	Note di rilascio
11 novembre 2025	Nuove funzionalità Amazon DocumentDB offre ora la piena compatibilità del protocollo cablato con MongoDB 8.0. Amazon DocumentDB 8.0 migliora le prestazioni delle query consentendoti di creare applicazioni ad alte prestazioni a un costo inferiore. Amazon DocumentDB 8.0 (versione della patch del motore 4.0.4448) • È stato aggiunto il supporto per i driver API MongoDB 8.0. • È stato aggiunto il supporto per Planner Version3 che estende i miglioramenti delle prestazioni agli operatori della fase di aggregazione, oltre a supportare le ottimizzazioni della pipeline di aggregazione e comandi distinti. • È stato aggiunto il supporto per la compressione basata sul dizionario tramite l'algoritmo di compressione Zstandard, migliorando il rapporto di compressione di 5 volte. • Nuove funzionalità aggiunte: Amazon DocumentDB 8.0 supporta la raccolta e le visualizzazioni. • È stato aggiunto il supporto per nuove fasi e operatori di aggregazione: $replaceWith, $vectorSearch, $merge, $set, $unset, $bucket e 3 nuovi operatori di aggregazione $pow, $rand, $DateTrunc. • È stato aggiunto il supporto per una nuova versione di text index: Text index v2 in Amazon DocumentDB 8.0 introduce token aggiuntivi, migliorando le funzionalità di ricerca testuale. • Include miglioramenti alla ricerca vettoriale attraverso la creazione parallela di indici vettoriali che riducono i tempi di creazione dell'indice fino a 30 volte. • È stato rimosso il supporto per TLS 1.0 e TLS 1.1. La versione minima di TLS è ora TLS 1.2.

Amazon DocumentDB 5.0

Note di rilascio per la versione 5.0 del motore Amazon DocumentDB (compatibilità con MongoDB 5.0).

Amazon DocumentDB 5.0.2

Data	Note di rilascio
28 settembre 2026	Amazon DocumentDB 5.0.2 (versione del motore: 3.0.23065) Nuove funzionalità e miglioramenti 1. È stato aggiunto il supporto per gli indici nascosti per tutti i tipi di indice (Planner versione 2 e successive). 2. Supporto per i metodi del `max()` cursore `min()` e del cursore sul comando find. 3. È stato aggiunto il supporto per il filtro degli indici in tutti i comandi. 4. Il pianificatore di query ora dà la priorità agli indici univoci per i predicati di quality (). `$eq` 5. È stato aumentato il limite di troncamento per i comandi aggregati in output. `currentOp` Correzioni di bug 1. È stato corretto un bug nella versione 1 (predefinita) del pianifica tore di query. 2. È stata corretta la gestione parziale del testo dell'espressione dell'indice durante il Major Version Upgrade (MVU). 3. È stato risolto un problema relativo al profiler per le query di conteggio rapido.

Amazon DocumentDB 5.0.1

Data	Note di rilascio
18 agosto 2026	Miglioramenti della sicurezza e della stabilità Amazon DocumentDB 5.0.1 (versione del motore: 3.0.22319) [in corso] Questa patch del motore offre i seguenti miglioramenti: • Una migliore elaborazione delle richieste Amazon DocumentDB ora convalida le richieste dei client in arrivo in modo più accurato lungo il percorso di connessione e analisi dei messaggi. Ciò migliora la resilienza e la stabilità in una gamma più ampia di modelli di input. • Il rafforzamento dell'handshake di autenticazione Amazon DocumentDB ora convalida lo scambio di autenticazione in modo più completo. Ciò migliora la solidità nella gestione di input imprevist i o non validi. Se non la applichi prima, Amazon DocumentDB applica automatic amente questa patch durante la prossima finestra di manutenzione, 30 giorni dopo il rilascio.
11 agosto 2026	Nuova funzionalità Amazon DocumentDB 5.0.1 (versione del motore: 3.0.21138) Supporto abilitato per l'autenticazione. SCRAM-SHA-256
8 giugno 2026	Nuova versione minore Amazon DocumentDB 5.0.1 (versione del motore 3.0.21138) Amazon DocumentDB 5.0.1 è la prima versione secondaria della versione principale di Amazon DocumentDB 5.0. Le versioni minori includono miglioramenti come nuove funzionalità, correzioni di sicurezza e correzioni di bug. È possibile eseguire l'aggiornamento alla versione 5.0.1 dalla versione 5.0.0 utilizzando un aggiornam

Data	Note di rilascio
	ento della versione secondaria. Per ulteriori informazioni, consulta Aggiornamento della versione secondaria di Amazon DocumentDB. Questa versione minore include le seguenti nuove funzionalità e miglioramenti: 1. `gcRuntimeStats` Aggiunta `documentFragmentStats` e suddivisione delle metriche a livello di raccolta `collstats` per fornire ulteriori informazioni sullo stato generale del processo di raccolta dei rifiuti. `storageSegmentBase` `storageSegmentExtended` Per informazioni dettagliate, vedi Raccolta dei rifiuti in Amazon DocumentDB. 2. Supporto per gli operatori di aggregazione e supporto`$rand`. `$pow` 3. Ora puoi monitorare la `Connections (active)` metrica per le istanze. Per ulteriori informazioni, consulta Quote delle istanze. 4. Migliore resilienza interregionale dei cluster globali eliminando il riavvio secondario del cluster con Global Survivable Read Only. 5. Supporto e operatori di aggregazione. `$dateToParts` `$dateFromParts` 6. Command-level le metriche delle prestazioni sono ora disponibili CloudWatch per le operazioni CRUD (`find`,`insert`,`findAndModify` ,,`update`,`delete`,`aggregate` ,`count`,`distinct`,`commitTransaction`) `getMoreabortTransaction` , l'esposizione `AvgDuration` e `MaxConcurrent` le dimensioni per un `P100Duration` migliore monitoraggio operativo. 7. `$replaceRoot` restituisce `newRoot` campi contenenti valori booleani (`true`/`false`) e valori numerici (/). `0` `1` 8. `$indexOfArray` supporta una gamma più ampia di espressioni di interrogazione valide. 9. Il piano di spiegazione include le `docsExamined` metriche nell'`executionStats` output. 10. Prestazioni di autenticazione SASL migliorate.

Amazon DocumentDB 5.0.0

Data	Note di rilascio
18 agosto 2026	Miglioramenti della sicurezza e della stabilità Amazon DocumentDB 5.0.0 (versione del motore: 3.0.22303) [in corso] Questa patch del motore offre i seguenti miglioramenti: • Una migliore elaborazione delle richieste Amazon DocumentDB ora convalida le richieste dei client in arrivo in modo più accurato lungo il percorso di connessione e analisi dei messaggi. Ciò migliora la resilienza e la stabilità in una gamma più ampia di modelli di input. • Il rafforzamento dell'handshake di autenticazione Amazon DocumentDB ora convalida lo scambio di autenticazione in modo più completo. Ciò migliora la solidità nella gestione di input imprevist i o non validi. • Access-control applicazione Amazon DocumentDB ora verifica i privilegi richiesti in modo coerente a tutti i livelli di nidificazione per le operazioni di query annidate. Se non la applichi prima, Amazon DocumentDB applica automatic amente questa patch durante la prossima finestra di manutenzione, 30 giorni dopo il rilascio.
6 maggio 2026	Correzioni di bug e altre modifiche Amazon DocumentDB 5.0 (versione del motore: 3.0.20421) Questa versione riguarda una patch di sicurezza richiesta.
20 aprile 2026	Nuova funzionalità In-place L'aggiornamento della versione principale (MVU) da Amazon DocumentDB 5.0 ad Amazon DocumentDB 8.0 è ora disponibile. Per

Data	Note di rilascio
	ulteriori informazioni, consulta Aggiornamento della versione principale in-place di Amazon DocumentDB.
13 aprile 2026	Correzioni di bug e altre modifiche Amazon DocumentDB 5.0 (versioni del motore: 3.0.19917) Questa è una patch del motore obbligatoria che include le seguenti nuove funzionalità, miglioramenti e correzioni: 1. La `DatabaseConnectionsActiveMax` metrica ora tiene traccia delle `Connections (active)` istanze. Per ulteriori informazioni, consulta Quote delle istanze. 2. Prestazioni di autenticazione SASL migliorate. Tutti i cluster Amazon DocumentDB 5.0 verranno aggiornati automatic amente a questa versione di patch durante la prossima finestra di manutenzione.

Data	Note di rilascio
22 ottobre 2025	Nuove funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.17983) Amazon DocumentDB ora supporta l'ultima generazione di istanze ottimizzate per la memoria, db.r8g, alimentate da processori Arm-based AWS Graviton4, che forniscono prestazioni fino al 30% migliori rispetto alle istanze R6G. Per ulteriori Gestione delle classi di istanze informazioni, consulta questa guida e questo post del blog. Nuove funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.15902) • È stato aggiunto il supporto per nomi di raccolta più lunghi (lunghezza dello spazio dei nomi fino a 255 caratteri). • È stato introdotto un nuovo pianificatore di query (Planner versione 2.0). Per ulteriori informazioni sul tagging, consulta Pianificatore di query v2in questa guida. • Sono stati aggiunti nuovi campi `collstats` per «mvccidStats» e «gcRuntimeStats" che forniscono informazioni sullo stato generale della raccolta rispetto alla raccolta dei rifiuti e all'invecchiamento delle raccolte.
16 ottobre 2025	Nuova funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.15902) Amazon DocumentDB ora supporta la modalità dual-stack (IPv4/IPv6) quando il cluster si trova in un cloud privato virtuale (VPC). Per ulteriori informazioni, consulta questo post del blog e la Amazon DocumentDB Amazon VPC e Amazon DocumentDB Developer Guide.

Data	Note di rilascio
15 settembre 2025	Correzioni di bug e altre modifiche Amazon DocumentDB 5.0 (versione della patch del motore 3.0.17983) Risolto il problema del crash del motore che si verificava quando le query contenevano una fase successiva. `$replaceRoot` `$lookup`

Data	Note di rilascio
29 luglio 2025	Nuove funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.15902) • È stato aggiunto il supporto dell'operatore di aggregazione. `$regexFindAll` • Lancio delle metriche di ingrandimento dell'indice e basate sulla raccolta • Sono state aggiunte CloudWatch metriche `AvailableMVCCIds` `LongestActiveGCRuntime` che forniscono informazioni sullo stato generale del cluster per quanto riguarda la raccolta dei rifiuti e la raccolta obsoleta. • Il `serverStatus` comando ora include un campo «nvme_cac he» che mostra il numero di pagine scritte e non scritte nella cache NVMe sulle istanze. NVMe-backed • È stata rimossa la limitazione al numero di campi per e `$group` `$project` • Supporto per i seguenti operatori di stringa: `$trim` `$rtrim`,`$ltrim`,`$regexFindAll` ,`$replaceOne` , e `$replaceAll` • Supporto per i seguenti operatori Array: `$first` e `$last` • Supporto per il seguente operatore Stage: `$collStats` Correzioni di bug e altre modifiche Amazon DocumentDB 5.0 (versione della patch del motore 3.0.15902) È stato risolto un problema durante la creazione dell'indice che interessava raccolte di grandi dimensioni.

Data	Note di rilascio
28 luglio 2025	Nuove funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.14228) I cluster basati su istanze Amazon DocumentDB ora supportano Amazon DocumentDB Serverless, una configurazione su richiesta con scalabilità automatica. Per ulteriori informazioni, consulta questo post del blog What's new. Utilizzo di Amazon DocumentDB serverless https://aws.amazon.com/about-aws/whats-new/2025/07/amazon-documentdb-serverless/
24 marzo 2025	Nuova funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.11800) È stato aggiunto il supporto per i flussi di modifica in`postBatch ResumeToken` . Per ulteriori informazioni, consulta Ripresa di un flusso di modifiche con post `BatchResumeToken` .

Data	Note di rilascio
15 gennaio 2025	Nuove funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.11800) Amazon DocumentDB ora mantiene la disponibilità di lettura tramite il riavvio dell'istanza di scrittura. Le istanze Reader ora continueranno a fornire richieste di lettura durante i riavvii delle istanze di writer. Correzioni di bug e altre modifiche Amazon DocumentDB 5.0 (versione della patch del motore 3.0.11800) • `killOp`Comando fisso per gestire un caso speciale di inserimenti in blocco. • Migliore utilizzo della I/O-related memoria di rete sulle istanze Amazon DocumentDB. • `count`Comando fisso per le query sui `$text` filtri.
18 dicembre 2024	Nuove funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.5721) I cluster basati su istanze Amazon DocumentDB ora supportano le istanze R6GD. NVMe-backed Per ulteriori informazioni, consulta questo post del blog What's NewNVMe-backed istanze. https://aws.amazon.com/about-aws/whats-new/2024/12/nvme-backed-r6gd-instances-amazon-documentdb-mongodb-compatibility/

Data	Note di rilascio
6 novembre 2024	Nuove funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.11051) • È stato aggiunto il supporto per le versioni `tls1.2+` minime di TLS e per i parametri del cluster. `tls1.3+` `tls` • È stato abilitato il supporto per i caratteri dollaro ($) e punto (.) nei nomi dei campi. Per le differenze funzionali, vedereDollaro ($) e punto (.) nei nomi dei campi.
17 settembre 2024	Nuove funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.10696) Amazon DocumentDB ora supporta la ricostruzione degli indici con. `reIndex` `runCommand` Per ulteriori informazioni, consulta Manutenzione dell'indice utilizzando ReIndex . Note `reIndex`è supportato solo su Amazon DocumentDB 5.0 (Engine Patch versione 3.0.10696 e successive). Correzioni di bug e altre modifiche Amazon DocumentDB 5.0 (versione della patch del motore 3.0.10696) • `$setOnInsert` ora supporta il `_id` campo durante gli inserimenti risultanti dalle operazioni di aggiornamento. • È stato risolto un problema che impediva il recupero dello spazio di archiviazione dopo un importante aggiornamento della versione ad Amazon DocumentDB 5.0.

Data	Note di rilascio
22 agosto 2024	Nuova funzionalità Amazon DocumentDB 5.0 (tutte le versioni delle patch del motore) I cluster globali di Amazon DocumentDB ora supportano lo switchover dei cluster e i failover gestiti dei cluster. Per ulteriori informazioni, consulta Esecuzione di uno switchover per un cluster globale Amazon DocumentDB e Esecuzione di un failover gestito per un cluster globale Amazon DocumentDB. **Note** Gli switchover e i failover globali dei cluster sono supportati solo su Amazon DocumentDB 4.0 e 5.0.

<table>
<tr><th>Data</th><th>Note di rilascio</th></tr>
<tr><td>23 luglio 2024</td><td>
Nuove funzionalità

Amazon DocumentDB 5.0 (versione 3.0.8126 della patch del motore)

• È stato aggiunto il supporto per nomi di indice più lunghi (fino a 255 caratteri). Per ulteriori informazioni, consulta Vincoli per la denominazione
• Il limite massimo di connessione è stato raddoppiato:

<table>
<tr><th>Tipo di istanza</th><th>Limite originale</th><th>Nuovo limite</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>
<tr><td>r6g.4xlarge</td><td>14200</td><td>28400</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Data</th><th>Note di rilascio</th></tr>
<tr><td></td><td>
<table>
<tr><th>Tipo di istanza</th><th>Limite originale</th><th>Nuovo limite</th></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>
Per ulteriori informazioni, consulta Quote delle istanze.
Correzioni di bug e altre modifiche
Amazon DocumentDB 5.0 (versione della patch del motore 3.0.8126)
È stata migliorata la logica di sincronizzazione <code>CurrentTime</code> e il flusso di modifiche <code>ResumeToken</code> sui lettori.
</td></tr>
</table>

Data	Note di rilascio

22 luglio 2024	Nuove funzionalità
	
	Amazon DocumentDB 5.0 (versione della patch del motore 3.0.6742)
	
	• È stato aggiunto il supporto per il filtro DML Audit. Ora puoi impostare le condizioni di filtro per filtrare i log di controllo DML in base ai loro requisiti specifici invece di registrare ogni query DML. Per ulteriori informazioni, consulta Filtraggio degli eventi di controllo DML.
	• È stato aggiunto il supporto per la compressione dei documenti per quanto segue:
	  • Impostazione di una soglia minima di compressione
	  • Abilitazione della compressione per le raccolte esistenti (applicab ile ai nuovi documenti)
	  • Consenti l'impostazione di compressione predefinita a livello di cluster
	
	  Per ulteriori informazioni, consulta Gestione della compressione dei documenti a livello di raccolta.
	• È stato aggiunto il supporto per l'utilizzo dei flussi di modifiche sulle istanze di lettura. Per ulteriori informazioni, consulta Utilizzo dei flussi di modifica su istanze secondarie.

9 luglio 2024	Nuova funzionalità Per Amazon DocumentDB 5.0 (versione minima della patch del motore 3.0.4780), gli aggiornamenti dei certificati TLS CA non richiedono più il riavvio del sistema. Per ulteriori informazioni, consulta Aggiornamento dei certificati TLS di Amazon DocumentDB.

Data	Note di rilascio
29 maggio 2024	Nuove funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.6742) • È stato aggiunto il supporto per e gli operatori. `$regexMatch` `$regexFind` • È stato aggiunto il supporto per garantire la massima precision e nei log di controllo quando si indirizzano numeri interi di grandi dimensioni. I log di controllo ora mantengono l'esatta rappresen tazione numerica di tutti i numeri, prevenendo qualsiasi perdita di precisione.
3 aprile 2024	Nuove funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.5721) • È stato aggiunto il supporto `bypassDocumentValidation` e il messaggio di errore granulare per. `$jsonSchema` Per ulteriori informazioni su `bypassDocumentValidation` , consultare bypassare DocumentValidation. • È stato aggiunto il supporto di. `$expr` • È stato aggiunto il supporto per Uncorrelated Joins in. `$lookup` • È stato aggiunto il supporto per mantenere le regole di convalida in fase di aggregazione. `$out`

Data	Note di rilascio
10 gennaio 2024	Nuove funzionalità Amazon DocumentDB 5.0 (versioni della patch del motore 3.0.4574, 3.0.4780, 3.0.4960) • È stato aggiunto il supporto per gli indici vettoriali HNSW. Per ulteriori informazioni, consulta Ricerca vettoriale per Amazon DocumentDB. • È stato aggiunto il supporto per gli indici parziali. Per ulteriori informazioni, consulta Indice parziale. • È stato aggiunto il supporto per GC runtime su una raccolta all'inter no del comando. `current0p` • È stato aggiunto il supporto per l'indice di testo per la ricerca di testo nativo su Amazon DocumentDB. Per ulteriori informazioni, consulta Esecuzione di ricerche di testo con Amazon DocumentDB. • È stato aggiunto il supporto per le parole chiave `$jsonSchema` dello schema `type` `allOf` `oneOf` `anyOfnot`,`maxItems`,`minItems`,,`maxProperties` ,`minProper ties` ,`pattern`,`patternProperties` ,`multiple0 f` ,`dependencies` , `euniqueItems` . Per ulteriori informazioni, consulta Utilizzo della convalida dello schema JSON. • È stato aggiunto il supporto per gli operatori aritmetic i`$ceil`,,`$floor`,`$ln`, `$log``$log10`, `$sqrt` e. `$exp`  Per ulteriori informazioni, consulta Operatori aritmetici. • È stato aggiunto il supporto per l'operatore di espressione condizion ale. `$switch` • È stato aggiunto il supporto per la `IVFFLAT` compilazione parallela di indici vettoriali. La documentazione è stata aggiornata rimuovend o la limitazione delle build di indici `IVFFLAT` vettoriali paralleli dalla guida per gli sviluppatori.

Data	Note di rilascio
	Correzioni di bug e altre modifiche • È stata aggiunta la funzionalità di richiamo senza distinzione tra maiuscole e minuscole. `db.runCommand("dbstats")` I clienti di Amazon DocumentDB 5.0 con versioni di patch del motore precedenti alla 3.0.4960 o alla 2.0.10221 devono applicare queste ultime patch del motore. • È stato corretto un errore durante l'invocazione su mongo shell versione 1.7 e successive. `db.coll.stats()` La documenta zione è stata aggiornata rimuovendo il suggerimento per la `db.coll.stats()` risoluzione dei problemi di mongo shell dalla guida per gli sviluppatori.
29 novembre 2023	Nuove funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.3727) È stato aggiunto il supporto per la ricerca vettoriale. Per ulteriori informazioni, consulta questo post del blog e Ricerca vettoriale per Amazon DocumentDB consulta la Amazon DocumentDB Developer Guide.
21 novembre 2023	Nuove funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.3727) È stato aggiunto il supporto per lo storage. I/O-optimized Per ulteriori informazioni, consulta Configurazioni di storage del cluster Amazon DocumentDB la Amazon DocumentDB Developer Guide. È stata aggiunta l'integrazione per l'apprendimento automatico senza codice con SageMaker Canvas. Per ulteriori informazioni, consulta No-code apprendimento automatico con Amazon SageMaker AI Canvas la Amazon DocumentDB Developer Guide.

Data	Note di rilascio
6 novembre 2023	Amazon DocumentDB 5.0 (versioni della patch del motore 3.0.3727) Nuove funzionalità • È stato aggiunto il supporto per le parole chiave `$jsonSchema` dello schema `maxLength` ,`minLength` ,`maximum`,, `minimum` `exclusiveMaximum` , `exclusiveMinimum` e. `items` `additionalItems` Nota che la convalida dello schema JSON è supportata solo nei cluster basati su istanze. • È stato aggiunto il supporto per l'operatore `$convert` di pipeline di aggregazione e i relativi operatori `$toBool` derivati in forma abbreviata,,,,, e. `$toInt` `$toLong` `$toDouble` `$toString` `$toDecimal` `$toObjectId` `$toDate` • È stato aggiunto il supporto per gli operatori `$setDifference` di espressione impostata,, e. `$anyElementTrue` `$allElementTrue` Correzioni di bug e altre modifiche È stato risolto il problema per cui non `NaN` veniva visualizzato un aggiornamento del flusso di modifiche da `-NaN` a.
15 settembre 2023	Nuove funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.3140) È stato aggiunto il supporto per il validatore dello schema $JsonSchema solo nei cluster basati su istanze. Per ulteriori informazioni, consulta Utilizzo della convalida dello schema JSON.

Data	Note di rilascio
13 luglio 2023	Nuove funzionalità Amazon DocumentDB 5.0 (versione della patch del motore 3.0.1948) • È stato aggiunto il supporto per la compressione dei documenti. • È stato aggiunto il supporto per la compilazione parallela di indici. • È stato aggiunto il supporto per lo stato di creazione dell'indice. Correzioni di bug e altre modifiche Amazon DocumentDB 5.0 (versione della patch del motore 3.0.1948) • È stato risolto il problema di autenticazione `createCollection` per i cluster elastici di Amazon DocumentDB quando gli utenti non hanno accesso alle raccolte di sistema. • È stato risolto il problema per cui le istanze della regione secondari a non potevano utilizzare gli stessi nomi di istanze della regione primaria.
7 giugno 2023	Correzioni di bug e altre modifiche Amazon DocumentDB 5.0 • Le istanze r5 e t3.medium sono ora supportate in Amazon DocumentDB 5.0. • `engineVersion` l'opzione predefinita è `5.0.0` in AWS SDK,, e. AWS CLI CloudFormation

Data	Note di rilascio
10 maggio 2023	Correzioni di bug e altre modifiche Amazon DocumentDB 5.0 (versione della patch del motore 3.0.1361) • È stato aggiunto il supporto per `ignoreunknownindexoptions` in the command. createIndex • Ho interrotto l'aggiunta di interrogazioni di monitoraggio interne ai registri di controllo. • I nomi delle variabili locali definiti dall'utente ora supportano «_» (trattino basso) per operatori di proiezione come e. `$let` `$filter`

Data	Note di rilascio

1 marzo 2023	Nuove funzionalità
	
	Amazon DocumentDB 5.0 (Engine Patch versione 3.0.775)
	
	• È stato introdotto Amazon DocumentDB 5.0
	  • Compatibilità con MongoDB 5.0 (supporto per i driver API MongoDB 5.0)
	  • Supporto per la crittografia a livello di Client-side campo (FLE). Ora puoi crittografare i campi sul lato client prima di scrivere i dati nel cluster Amazon DocumentDB. Per ulteriori informazioni, consulta Crittografia a livello di campo Client-side
	  • Nuovi operatori di aggregazione:`$dateAdd`, `$dateSubtract`
	• Limite di archiviazione aumentato a 128 TiB per tutti i cluster Amazon DocumentDB basati su istanze e i cluster elastici basati su shard.
	• Amazon DocumentDB 5.0 ora supporta la scansione dell'indice con l'operatore nel primo livello di nesting. `$elemMatch` Le scansioni degli indici sono supportate quando le query hanno un solo livello di `$elemMatch` filtro e la `$elemMatch` query annidata non supporta la scansione dell'indice.
	
	Forma di interrogazione che supporta la scansione dell'indice:
	
	`db.foo.find( { "a": {$elemMatch: { "b": "xyz", "c": "abc"} } })`
	
	Forma di interrogazione che non supporta la scansione dell'indice:
	
	`db.foo.find( { "a": {$elemMatch: { "b": {$elemMatch: { "d": "xyz", "e": "abc"} }} } })`

Amazon DocumentDB 4.0

Note sulla versione 4.0 del motore Amazon DocumentDB (compatibilità con MongoDB 4.0).

Data	Note di rilascio
31 agosto 2026	Nuova funzionalità Amazon DocumentDB ora supporta gli aggiornamenti delle versioni principali (MVU) sul posto direttamente dalla versione 4.0 del motore alla versione 8.0. Ciò elimina la necessità di aggiornamenti delle versioni intermedie e consente di aggiornare i cluster 4.0 preservan do i dati, le configurazioni e le impostazioni dei cluster esistenti. Questo aggiornamento è disponibile in tutte le AWS regioni in cui queste versioni sono attualmente supportate. Per ulteriori informazioni, consulta Aggiornamento della versione principale in-place di Amazon DocumentDB.
18 agosto 2026	Miglioramenti della sicurezza e della stabilità Amazon DocumentDB 4.0 (versione del motore: 2.0.14287) [in corso] Questa patch del motore offre i seguenti miglioramenti: • Una migliore elaborazione delle richieste Amazon DocumentDB ora convalida le richieste dei client in arrivo in modo più accurato lungo il percorso di connessione e analisi dei messaggi. Ciò migliora la resilienza e la stabilità in una gamma più ampia di modelli di input. • Il rafforzamento dell'handshake di autenticazione Amazon DocumentDB ora convalida lo scambio di autenticazione in modo più completo. Ciò migliora la solidità nella gestione di input imprevist i o non validi. • Access-control applicazione Amazon DocumentDB ora verifica i privilegi richiesti in modo coerente a tutti i livelli di nidificazione per le operazioni di query annidate. Se non la applichi prima, Amazon DocumentDB applica automatic amente questa patch durante la prossima finestra di manutenzione, 30 giorni dopo il rilascio.
27 aprile 2026	Correzioni di bug e altre modifiche

Data	Note di rilascio
	Amazon DocumentDB 4.0 (versione del motore: 2.0.14278) Questa è una patch del motore richiesta che risolve i seguenti problemi: 1. È stato risolto un problema che poteva, in rari casi, far sì che il ripristino point-in-time (PITR) e gli aggiornamenti delle versioni principali (MVU) non rispondessero. Tutti i cluster Amazon DocumentDB 4.0 verranno aggiornati automatic amente a questa versione di patch durante la prossima finestra di manutenzione.
9 marzo 2026	Correzioni di bug e altre modifiche Amazon DocumentDB 4.0 (versione del motore: 2.0.14165) Questo aggiornamento fornisce una patch opzionale del motore con i seguenti miglioramenti: 1. La `DatabaseConnectionsActiveMax` metrica ora tiene traccia delle `Connections (active)` istanze. Per ulteriori informazioni, consulta Quote delle istanze. 2. Prestazioni di autenticazione SASL migliorate.
22 ottobre 2025	Nuove funzionalità Amazon DocumentDB 4.0 (versione della patch del motore 2.0.11747) Sono stati aggiunti nuovi campi `collstats` per «mvccidStats» e «gcRuntimeStats" che forniscono informazioni sullo stato generale della raccolta rispetto alla raccolta dei rifiuti e all'invecchiamento delle raccolte.

Data	Note di rilascio
16 ottobre 2025	Nuova funzionalità Amazon DocumentDB 4.0 (versione della patch del motore 2.0.11747) Amazon DocumentDB ora supporta la modalità dual-stack (IPv4/IPv6) quando il cluster si trova in un cloud privato virtuale (VPC). Per ulteriori informazioni, consulta questo post del blog e la Amazon DocumentDB Amazon VPC e Amazon DocumentDB Developer Guide.
29 luglio 2025	Nuove funzionalità Amazon DocumentDB 4.0 (versione della patch del motore 2.0.11747) • Lancio delle metriche di ingrandimento dell'indice e del gonfiamento basate sulla raccolta • Sono state aggiunte CloudWatch metriche `AvailableMVCCIds` `LongestActiveGCRuntime` che forniscono informazioni sullo stato generale del cluster per quanto riguarda la raccolta dei rifiuti e la raccolta obsoleta. • Supporto per i seguenti operatori di stringa:`$trim`,, e `$rtrim` `$ltrim` • Supporto per il seguente operatore Stage: `$collStats` Correzioni di bug e altre modifiche Amazon DocumentDB 4.0 (versione della patch del motore 2.0.11747) È stato risolto un problema durante la creazione dell'indice che interessava raccolte di grandi dimensioni.

Data	Note di rilascio
15 gennaio 2025	Correzioni di bug e altre modifiche Amazon DocumentDB 4.0 (versione della patch del motore 2.0.11096) • `killOp`Comando fisso per gestire un caso speciale di inserimenti in blocco. • Migliore utilizzo della I/O-related memoria di rete sulle istanze Amazon DocumentDB.
6 novembre 2024	Nuove funzionalità Amazon DocumentDB 4.0 (versione della patch del motore 2.0.10980) • È stato aggiunto il supporto per le versioni `tls1.2+` minime di TLS e per i parametri del cluster. `tls1.3+` `tls`
17 settembre 2024	Correzioni di bug e altre modifiche Amazon DocumentDB 4.0 (versione della patch del motore 2.0.10898) • `$setOnInsert` ora supporta il `_id` campo durante gli inserimenti risultanti dalle operazioni di aggiornamento. • È stato risolto un problema che impediva il recupero dello spazio di archiviazione dopo un importante aggiornamento della versione ad Amazon DocumentDB 5.0.

Data	Note di rilascio
22 agosto 2024	Nuova funzionalità Amazon DocumentDB 4.0 (Engine Patch versione 2.0.5704) I cluster globali di Amazon DocumentDB ora supportano gli switchover dei cluster e i failover gestiti dei cluster. Per ulteriori informazioni, consulta Esecuzione di uno switchover per un cluster globale Amazon DocumentDB e Esecuzione di un failover gestito per un cluster globale Amazon DocumentDB. **Note** Gli switchover e i failover globali dei cluster sono supportati solo su Amazon DocumentDB 4.0 e 5.0.

<table>
<tr><th>Data</th><th>Note di rilascio</th></tr>
<tr><td>23 luglio 2024</td><td>
Nuove funzionalità

Amazon DocumentDB 4.0 (versione della patch del motore 2.0.10709)

• È stato aggiunto il supporto per nomi di indice più lunghi (fino a 255 caratteri). Per ulteriori informazioni, consulta Vincoli per la denominazione
• Il limite massimo di connessione è stato raddoppiato:

<table>
<tr><th>Tipo di istanza</th><th>Limite originale</th><th>Nuovo limite</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>
<tr><td>r6g.4xlarge</td><td>14200</td><td>28400</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Data</th><th>Note di rilascio</th></tr>
<tr><td></td><td>
<table>
<tr><th>Tipo di istanza</th><th>Limite originale</th><th>Nuovo limite</th></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>
Per ulteriori informazioni, consulta Quote delle istanze.
</td></tr>
<tr><td>22 luglio 2024</td><td>Nuove funzionalità

Amazon DocumentDB 4.0 (versione della patch del motore 2.0.10593)

• È stato aggiunto il supporto per il filtro DML Audit. Ora puoi impostare le condizioni di filtro per filtrare i log di controllo DML in base ai loro requisiti specifici invece di registrare ogni query DML. Per ulteriori informazioni, consulta Filtraggio degli eventi di controllo DML.</td></tr>
<tr><td>9 luglio 2024</td><td>Nuova funzionalità

Per Amazon DocumentDB 4.0 (versione minima della patch del motore 2.0.10179), gli aggiornamenti dei certificati TLS CA non richiedono più il riavvio del sistema. Per ulteriori informazioni, consulta Aggiornamento dei certificati TLS di Amazon DocumentDB.</td></tr>
<tr><td>29 maggio 2024</td><td>Nuove funzionalità

Amazon DocumentDB 4.0 (versione della patch del motore 2.0.10593)

• È stato aggiunto il supporto per garantire la massima precision e nei log di controllo quando si indirizzano numeri interi di grandi dimensioni. I log di controllo ora mantengono l'esatta rappresen tazione numerica di tutti i numeri, prevenendo qualsiasi perdita di precisione.</td></tr>
</table>

Data	Note di rilascio
3 aprile 2024	Nuove funzionalità Amazon DocumentDB 4.0 (versione della patch del motore 2.0.10392) • È stato aggiunto il supporto per. `bypassDocumentValidation` `$jsonSchema` Per ulteriori informazioni su `bypassDocumentValidation` , consultare bypassare DocumentValidation. • È stato aggiunto il supporto di`$expr`. • È stato aggiunto il supporto per Uncorrelated Joins in. `$lookup` • È stato aggiunto il supporto per mantenere le regole di convalida in fase di aggregazione. `$out`
10 gennaio 2024	Nuove funzionalità Amazon DocumentDB 4.0 (versioni della patch del motore 2.0.10124, 2.0.10179, 2.0.10221) • È stato aggiunto il supporto per GC runtime su una raccolta all'interno del comando. `currentOp` • È stato aggiunto il supporto per le parole chiave `$jsonSchema` dello schema `type` `allOf` `oneOf` `anyOf` `not`,`maxItems`,`minItems`,,`maxProperties` ,`minProperties` ,`pattern`,`patternProperties` ,`multipleOf` ,`dependencies` , `uniqueItems` . Per ulteriori informazioni, consulta Utilizzo della convalida dello schema JSON. • È stato aggiunto il supporto per gli operatori aritmetici `$ceil`,,`$floor`,`$ln`, `$log` `$log10`, `$sqrt` e. `$exp` Per ulteriori informazioni, consulta Operatori aritmetici. • È stato aggiunto il supporto per l'operatore di espressione condizionale. `$switch`

Data	Note di rilascio
20 dicembre 2023	Altre modifiche Supporto abilitato per l'aggiornamento diretto della versione principal e in Amazon DocumentDB 4.0. Per ulteriori informazioni, consulta Aggiornamento della versione principale in-place di Amazon DocumentDB.
6 novembre 2023	Amazon DocumentDB 4.0 (versioni della patch del motore 2.0.9876) Nuove funzionalità • È stato aggiunto il supporto per le parole chiave `$jsonSchema` dello schema `maxLength` ,`minLength` ,`maximum`,, `minimum` `exclusiveMaximum` , `exclusiveMinimum` e. `items` `additionalItems` Nota che la convalida dello schema JSON è supportata solo nei cluster basati su istanze. • È stato aggiunto il supporto per l'operatore `$convert` di pipeline di aggregazione e i relativi operatori `$toBool` derivati abbreviati,,,,, e. `$toInt` `$toLong` `$toDouble` `$toString` `$toDecimal` `$toObjectId` `$toDate` • È stato aggiunto il supporto per gli operatori `$setDifference` di espressione impostata,, e. `$anyElementTrue` `$allElementTrue` Correzioni di bug e altre modifiche È stato risolto il problema per cui non `NaN` veniva visualizzato un aggiornamento del flusso di modifiche da `-NaN` a.

Data	Note di rilascio
20 settembre 2023	Nuove funzionalità È stato aggiunto il supporto per gli aggiornamenti in loco delle principal i versioni in Amazon DocumentDB 4.0. Per ulteriori informazioni, consulta Aggiornamento della versione principale in-place di Amazon DocumentDB.
15 settembre 2023	Nuove funzionalità Amazon DocumentDB 4.0 (Engine Patch versione 2.0.9686) È stato aggiunto il supporto per il validatore dello schema $JsonSche ma solo nei cluster basati su istanze. Per ulteriori informazioni, consulta Utilizzo della convalida dello schema JSON.
13 luglio 2023	Nuove funzionalità Amazon DocumentDB 4.0 (versione della patch del motore 2.0.9259) • È stato aggiunto il supporto per la compilazione parallela di indici. Correzioni di bug e altre modifiche Amazon DocumentDB 4.0 (versione della patch del motore 2.0.9259) • Ha smesso di aggiungere query di monitoraggio interne ai log di controllo.

Data	Note di rilascio
4 aprile 2023	Correzioni di bug e altre modifiche Amazon DocumentDB 4.0 (versione della patch del motore 2.0.8934) • È stato risolto il problema del controllo DML quando veniva abilitato durante un carico di lavoro in corso. • È stato risolto un problema relativo al controllo DML quando ai comandi aggregati con hint viene passato un valore di stringa. • È stato risolto il problema relativo al `listCollections` comando che non funzionava quando gli utenti con ruolo readwriteanydataba se avevano entrambe le opzioni AuthorizedCollections e NameOnly impostate su true. • È stato risolto il problema relativo all'analisi corretta della stringa numerica in un nome di campo. • Annulla i cursori che scorrono a lungo quando influiscono sulla Garbage Collection. • I nomi delle variabili locali definiti dall'utente ora supportano «_» (trattino basso) per operatori di proiezione come e. `$let` `$filter`
27 febbraio 2023	Correzioni di bug e altre modifiche Amazon DocumentDB 4.0 È stato aggiunto il supporto per AWS Lambda. Per ulteriori informazi oni, vedere Utilizzo AWS Lambda con Change Streams.

Data	Note di rilascio
9 agosto 2022	Nuove funzionalità 4.0 • È stato aggiunto il supporto per il tipo di dati Decimal128. Il Decimal128 è un tipo di dati BSON supportato in tutte le regioni in cui DocumentDB è disponibile. Per ulteriori informazioni, consulta la sezione relativa ai tipi di dati. • È stato aggiunto il supporto per il controllo delle query DML con Amazon Logs. CloudWatch Ora Amazon DocumentDB è in grado di registrare eventi DML (Data Manipulation Language) ed eventi Data Definition Language (DDL) su Amazon Logs. CloudWatch Per ulteriori informazioni, consulta questo post del blog. Correzioni di bug e altre modifiche 4.0 • È ora possibile modificare la propria password con una propria password con privilegio. `changeOwnPassword`
25 luglio 2022	Nuove funzionalità Amazon DocumentDB 4.0 Ora puoi creare cluster più velocemente grazie alla possibilità di creare cloni che utilizzano lo stesso volume del cluster DocumentDB e hanno gli stessi dati del cluster originale. Per informazioni dettagliate, consulta Gestione dei cluster Amazon DocumentDB.

Data	Note di rilascio
27 giugno 2022	Nuove funzionalità Amazon DocumentDB 4.0 (versione della patch del motore 2.0.7509) Amazon DocumentDB ridimensiona dinamicamente il database in base ai modelli di utilizzo. L'aggiunta di altri dati aumenta lo spazio fino a 64 Tebibyte (TiB) e l'eliminazione dei dati riduce lo spazio assegnato .
7 aprile 2022	Nuove funzionalità 4.0 (versioni Engine Patch 2.0.6142 e 2.0.6948) Amazon DocumentDB Performance Insights è ora disponibile in anteprima. Ora puoi archiviare sette giorni di cronologia delle prestazioni in una finestra continua senza costi aggiuntivi. Per ulteriori informazioni, consulta Monitoring with Performance Insights.

24 gennaio 2022	Nuove funzionalità
	
	4.0 (Engine Patch versione 2.0.5170)
	
	• Amazon DocumentDB offre ora una versione di prova gratuita. Per ulteriori informazioni, consulta la pagina di prova gratuita di Amazon DocumentDB.
	• Ora puoi utilizzare funzionalità avanzate con Geospatial Query, incluse le seguenti API:
	  • `$geoWithin`
	  • `$geoIntersects`
	• È stato aggiunto il supporto per i seguenti operatori MongoDB:
	  • `$mergeObjects`
	  • `$reduce`
	
	Per ulteriori informazioni, consulta la sezione Interrogazione di dati geospaziali con Amazon DocumentDB.

Data	Note di rilascio

21 gennaio 2022	Nuove funzionalità
	
	Amazon DocumentDB 4.0 (versione della patch del motore 2.0.5706)
	
	• Le istanze Amazon DocumentDB Graviton2 (r6g.large, r6g.2xlarge, r6g.4xlarge, r6g.8xlarge, r6g.12xlarge, r6g.16xlarge e t4g.medium) sono ora supportate.
	
	Nuove funzionalità
	
	4.0 (Engine Patch versione 2.0.5706)
	
	• È stato aggiunto il supporto per le seguenti API MongoDB:
	  • `$reduce`
	  • `$mergeObjects`
	  • `$geoWithin`
	  • `$geoIntersects`

Data	Note di rilascio

25 ottobre 2021	Nuove funzionalità
	
	4.0 (Engine Patch versione 2.0.5704)
	
	• È stato aggiunto il supporto per le seguenti API MongoDB
	  • `$literal`
	  • `$map`
	  • `$$ROOT`
	• Supporto per le funzionalità di interrogazione. GeoSpatial Vedi questo post del blog per maggiori dettagli
	• Supporto per il controllo degli accessi con ruoli definiti dall'utente. Vedi questo post del blog per maggiori dettagli
	• Driver JDBC Amazon DocumentDB per abilitare la connettività da strumenti di BI come Tableau e strumenti di interrogazione come SQL Workbench
	
	Correzioni di bug e altre modifiche
	
	4.0 (Engine Patch versione 2.0.5704)
	
	• Correzione di bug `$natural` per l'ordinamento corretto quando è presente un esplicito insieme a `.sort()` `$natural`
	• Correzione di bug per il flusso di modifiche con cui lavorare `$redact`
	• Correzione di bug `$ifNull` per lavorare con un array vuoto
	• Correzione di un problema relativo all'eccessivo consumption/ server arresto anomalo delle risorse quando un utente attualmente connesso viene eliminato o a quell'utente viene revocato il privilegio di un'attività in corso
	• Correzione di bug e controllo dei privilegi `listDatabase` `listCollection`
	• Correzione di bug: logica di deduplicazione per elementi con più chiavi

Data	Note di rilascio
24 giugno 2021	Nuove funzionalità 4.0 (Engine Patch versione 2.0.3371) • Le istanze r5.8xlarge e r5.16xlarge sono ora supportate. Scopri di più nel post sul blog Amazon DocumentDB Now Supports r5.8xlarge e r5.16xlarge. • I cluster globali sono ora supportati per fornire il disaster recovery in caso di Region-wide interruzioni e consentire letture globali a bassa latenza consentendo le letture dal cluster Amazon DocumentDB più vicino. Tieni presente che i cluster globali non sono attualmente supportati nelle regioni di Sud America (San Paolo), Europa (Milano), Cina (Pechino) e Cina (Ningxia).

Data	Note di rilascio
04 maggio 2021	Nuove funzionalità 4.0 (Engine Patch versione 2.0.3371) • `renameCollection` • `$zip` • `$indexOfArray` • `$reverseArray` • `$natural` • `$hint`supporto per l'aggiornamento • Scansione dell'indice per `distinct` Correzioni di bug e altre modifiche 4.0 (Engine Patch versione 2.0.3371) • Utilizzo ridotto della memoria per le interrogazioni `$in` • È stata corretta una perdita di memoria negli indici a più chiavi • È stato corretto il piano di spiegazione e l'output del profiler per `$out` • È stato aggiunto un timeout per le operazioni dal sistema di monitoraggio interno per migliorare l'affidabilità • È stato corretto un difetto che influiva sui predicati delle query passati agli indici a più chiavi
15 gennaio 2021	Nuove funzionalità Amazon DocumentDB 4.0 (versione della patch del motore 2.0.722) • Nessuno

Data	Note di rilascio
9 novembre 2020	Nuove funzionalità Amazon DocumentDB 4.0 (versione della patch del motore 2.0.722) • Compatibilità con MongoDB 4.0 • Transazioni ACID • Supporto per `cluster(client.watch()` or `mongo.watch())` e i flussi di modifica a livello di `(db.watch())` database • Possibilità di avviare o riprendere un flusso di modifiche utilizzando `startAtOperationTime` • Estendi il periodo di conservazione del flusso di modifiche a 7 giorni (in precedenza 24 ore) • AWS DMS obiettivo per Amazon DocumentDB 4.0 • CloudWatch metriche:`TransactionsOpen` ,, `TransactionsOpenMax` `TransactionsAborted` , e `TransactionsStarted` `TransactionsCommitted` • Nuovi campi per le transazioni in`currentOp` ,`ServerStatus` , `eprofiler`. • Possibilità di utilizzare un indice con la fase di `$lookup` aggregazione • `find()`le interrogazioni con proiezioni possono essere fornite a partire da un indice (interrogazione coperta) • Possibilità di utilizzo `hint()` con `findAndModify` • Ottimizzazioni delle prestazioni per l'operatore `$addToSet` • Miglioramenti per ridurre le dimensioni complessive dell'indice. • Nuovi operatori di aggregazione: `$ifNull``$replaceRoot` ,`$setIsSubset` , `$setInstersection` `$setUnion` , e `$setEquals` • Con i `ListDatabase` comandi `ListCollection` and, ora è possibile utilizzare facoltativamente i `authorizedDatabases` parametri `authorizedCollections` and per consentire agli utenti di elencare le raccolte e i database a cui hanno l'autoriz

Data	Note di rilascio
	zazione di accesso senza richiedere rispettivamente i `listDatab ase` ruoli `listCollections` e • Gli utenti possono anche terminare i propri cursori senza richiedere il ruolo `KillCursor` • Il confronto di tipi numerici di documenti secondari è ora coerente con il confronto di tipi numerici di documenti di primo livello. Il comportamento in Amazon DocumentDB 4.0 è ora compatibile con MongoDB. Correzioni di bug e altre modifiche Amazon DocumentDB 4.0 (versione della patch del motore 2.0.722) • `$setOnInsert` non consentono più gli aggiornamenti quando si utilizza l'operatore posizionale. $ Il comportamento in Amazon DocumentDB 4.0 è ora compatibile con MongoDB. • Risolto il problema con and set `$createCollection autoIndexId` • Proiezione per documenti annidati • È stata modificata l'impostazione predefinita per la memoria di lavoro in base alla dimensione della memoria dell'istanza • Miglioramenti alla raccolta dei rifiuti • Ricerca con chiave vuota nel percorso, differenza di comportamento con mongo • Risolto un `dateToString` bug nel comportamento del fuso orario • Risolto `$push` (aggregazione) il rispetto dell'ordinamento • Risolto un bug `$currentOp` con aggregate • Risolto il problema relativo a un `readPreference` secondario • È stato risolto il problema relativo alla convalida dello stesso database in cui `$createIndex` è stato emesso il comando • È stato corretto un comportamento incoerente per`minKey`, `maxKey` la ricerca non riesce

Data	Note di rilascio
	• È stato risolto il problema relativo al `$size` mancato funzionamento dell'operatore con l'array composito • È stato risolto il problema relativo alla negazione di `$in` con regex • È stato risolto il problema relativo all'esecuzione del `$distinct` comando su una vista • È stato risolto il problema relativo alle aggregazioni e ai comandi di ricerca che ordinavano i campi mancanti in modo diverso • Risolto `$eq` il problema con l'espressione regolare che non controlla va il tipo • Risolto `$currentDate` un bug nel comportamento della posizione ordinale relativa al timestamp • Risolta la granularità in millisecondi per `$currentDate`

Amazon DocumentDB 3.6

Note di rilascio per la versione 3.6 del motore Amazon DocumentDB (compatibilità con MongoDB 3.6).

Data	Note di rilascio
31 agosto 2026	Nuova funzionalità Amazon DocumentDB ora supporta gli aggiornamenti delle versioni principali (MVU) sul posto direttamente dalla versione 3.6 alla versione 8.0 del motore. Ciò elimina la necessità di aggiornamenti di versione intermedi e consente di aggiornare i cluster 3.6 preservando i dati, le configurazioni e le impostazioni dei cluster esistenti. Questo aggiornamento è disponibile in tutte le AWS regioni in cui queste versioni sono attualmente supportate. Per ulteriori informazioni, consulta Aggiornamento della versione principale in-place di Amazon DocumentDB.
18 agosto 2026	Miglioramenti della sicurezza e della stabilità

Data	Note di rilascio
	Amazon DocumentDB 3.6 (versione del motore: 1.0.208702) [in corso] Questa patch del motore offre i seguenti miglioramenti: • Una migliore elaborazione delle richieste Amazon DocumentDB ora convalida le richieste dei client in arrivo in modo più accurato lungo il percorso di connessione e analisi dei messaggi. Ciò migliora la resilienza e la stabilità in una gamma più ampia di modelli di input. • Il rafforzamento dell'handshake di autenticazione Amazon DocumentDB ora convalida lo scambio di autenticazione in modo più completo. Ciò migliora la solidità nella gestione di input imprevisti o non validi. Se non la applichi prima, Amazon DocumentDB applica automaticamente questa patch durante la prossima finestra di manutenzione, 30 giorni dopo il rilascio.
27 aprile 2026	Nuove funzionalità Amazon DocumentDB 3.6 (versione del motore: 1.0.208692) Questo aggiornamento fornisce una patch del motore opzionale con i seguenti miglioramenti: 1. È stato aggiunto il supporto per `startAtOperationTime` i flussi di modifica.
12 febbraio 2026	Correzioni di bug e altre modifiche Amazon DocumentDB 3.6 (versione del motore: 1.0.208675) 1. La `DatabaseConnectionsActiveMax` metrica ora tiene traccia delle istanze. `Connections (active)` Per ulteriori informazioni, consulta Quote delle istanze. 2. Prestazioni di autenticazione SASL migliorate. 3. Correzione di bug per il comando $indexStats.

Data	Note di rilascio
20 agosto 2024	Nuova funzionalità Per Amazon DocumentDB 3.6 (versione minima della patch del motore 1.0.208662), gli aggiornamenti dei certificati TLS CA non richiedono più il riavvio del sistema. Per ulteriori informazioni, consulta Aggiornamento dei certificati TLS di Amazon DocumentDB.
22 luglio 2024	Nuove funzionalità Amazon DocumentDB 3.6 (versione della patch del motore 1.0.208662) Sono state rimosse le limitazioni agli indici sulle istanze db.r5.* e db.r6.* in Amazon DocumentDB MVU. Per ulteriori informazioni, consulta Prerequisiti. Correzioni di bug e altre modifiche Amazon DocumentDB 3.6 (versione della patch del motore 1.0.208662) Amazon DocumentDB ora riconosce -NaN come token JSON valido.
20 dicembre 2023	Altre modifiche Supporto abilitato per l'aggiornamento diretto della versione principale in Amazon DocumentDB 3.6. Per ulteriori informazioni, consulta Aggiornamento della versione principale in-place di Amazon DocumentDB.
17 novembre 2023	Correzioni di bug e altre modifiche Amazon DocumentDB 3.6 (versione della patch del motore 1.0.208570) I nomi delle variabili locali definiti dall'utente ora supportano «_» (trattino basso) per operatori di proiezione come e. `$let $filter`

Data	Note di rilascio
20 settembre 2023	Nuove funzionalità È stato aggiunto il supporto per gli aggiornamenti delle principali versioni in loco in Amazon DocumentDB 3.6. Per ulteriori informazioni, consulta Aggiornamento della versione principale in-place di Amazon DocumentDB.
2 febbraio 2023	Correzioni di bug e altre modifiche Amazon DocumentDB 3.6 (versione della patch del motore 1.0.20843 2) • È stato risolto il problema del controllo DML quando veniva abilitato durante un carico di lavoro in corso. • È stato risolto un problema relativo al controllo DML quando ai comandi aggregati con hint viene passato un valore di stringa. • È stato risolto il problema relativo al `listCollections` comando che non funzionava quando gli utenti con ruolo readwriteanydataba se avevano entrambe le opzioni AuthorizedCollections e NameOnly impostate su true. • È stato risolto il problema relativo all'analisi corretta della stringa numerica in un nome di campo. • Annulla i cursori che scorrono a lungo quando influiscono sulla Garbage Collection.

Data	Note di rilascio
9 agosto 2022	Nuove funzionalità Amazon DocumentDB 3.6 (versione della patch del motore 1.0.208152) • È stato aggiunto il supporto per il tipo di dati Decimal128. Il Decimal128 è un tipo di dati BSON supportato in tutte le regioni in cui DocumentDB è disponibile. Per ulteriori informazioni, consulta la sezione relativa ai tipi di dati. • È stato aggiunto il supporto per il controllo delle query DML con Amazon Logs. CloudWatch Ora Amazon DocumentDB è in grado di registrare eventi DML (Data Manipulation Language) ed eventi Data Definition Language (DDL) su Amazon Logs. CloudWatch Per ulteriori informazioni, consulta questo post del blog. Correzioni di bug e altre modifiche Amazon DocumentDB 3.6 (versione della patch del motore 1.0.208152) • Ora puoi cambiare la tua password con una password personale con privilegi. `changeOwnPassword`
7 aprile 2022	Nuove funzionalità Amazon DocumentDB 3.6 (versioni della patch del motore 1.0.207836 e 1.0.208015) Amazon DocumentDB Performance Insights è ora disponibile in anteprima. Ora puoi archiviare sette giorni di cronologia delle prestazioni in una finestra continua senza costi aggiuntivi. Per ulteriori informazioni, consulta Monitoring with Performance Insights.

Data	Note di rilascio

24 gennaio 2022	Nuove funzionalità
	
	Amazon DocumentDB 3.6 (versione della patch del motore 1.0.207684)
	
	• Amazon DocumentDB offre ora una versione di prova gratuita. Per ulteriori informazioni, consulta la pagina di prova gratuita di Amazon DocumentDB.
	• Ora puoi utilizzare funzionalità avanzate con Geospatial Query, incluse le seguenti API:
	  • `$geoWithin`
	  • `$geoIntersects`
	• È stato aggiunto il supporto per i seguenti operatori MongoDB:
	  • `$mergeObjects`
	  • `$reduce`
	
	Per ulteriori informazioni, consulta la sezione Interrogazione di dati geospaziali con Amazon DocumentDB.

21 gennaio 2022	Nuove funzionalità
	
	Amazon DocumentDB 3.6 (versione della patch del motore 1.0.207781)
	
	• È stato aggiunto il supporto per le seguenti API MongoDB:
	  • `$reduce`
	  • `$mergeObjects`
	  • `$geoWithin`
	  • `$geoIntersects`

Data	Note di rilascio

25 ottobre 2021	Nuove funzionalità
	
	Amazon DocumentDB 3.6 (versione della patch del motore 1.0.20778 0)
	
	• È stato aggiunto il supporto per le seguenti API MongoDB
	  • `$literal`
	  • `$map`
	  • `$$ROOT`
	• Supporto per le funzionalità di interrogazione. GeoSpatial Vedi questo post del blog per maggiori dettagli
	• Supporto per il controllo degli accessi con ruoli definiti dall'utente. Vedi questo post del blog per maggiori dettagli
	• Driver JDBC Amazon DocumentDB per abilitare la connettività da strumenti di BI come Tableau e strumenti di interrogazione come SQL Workbench
	
	Correzioni di bug e altre modifiche
	
	Amazon DocumentDB 3.6 (versione della patch del motore 1.0.20778 0)
	
	• Correzione di bug per l'ordinamento corretto quando `$natural` è presente un esplicito insieme a `.sort()` `$natural`
	• Correzione di bug per il flusso di modifiche con cui lavorare `$redact`
	• Correzione di bug `$ifNull` per lavorare con un array vuoto
	• Correzione di un problema relativo all'eccessivo consumption/ server arresto anomalo delle risorse quando un utente attualmente connesso viene eliminato o a quell'utente viene revocato il privilegio di un'attività in corso
	• Correzione di bug e controllo dei privilegi `listDatabase` `listCollection`

Data	Note di rilascio
	• Correzione di bug: logica di deduplicazione per elementi con più chiavi
24 giugno 2021	Nuove funzionalità Amazon DocumentDB 3.6 (versione della patch del motore 1.0.207117) • Le istanze r5.8xlarge e r5.16xlarge sono ora supportate. Scopri di più nel post sul blog Amazon DocumentDB Now Supports r5.8xlarge e r5.16xlarge. • I cluster globali sono ora supportati per fornire il disaster recovery in caso di Region-wide interruzioni e consentire letture globali a bassa latenza consentendo le letture dal cluster Amazon DocumentDB più vicino. Tieni presente che i cluster globali non sono attualmente supportati nelle regioni di Sud America (San Paolo), Europa (Milano), Cina (Pechino) e Cina (Ningxia).

Data	Note di rilascio
04 maggio 2021	Nuove funzionalità Amazon DocumentDB 3.6 (versione della patch del motore 1.0.20711 7) • `renameCollection` • `$zip` • `$indexOfArray` • `$reverseArray` • `$natural` • `$hint`supporto per l'aggiornamento • Scansione dell'indice per `distinct` Correzioni di bug e altre modifiche Amazon DocumentDB 3.6 (versione della patch del motore 1.0.20711 7) • Utilizzo ridotto della memoria per le query `$in` • È stata corretta una perdita di memoria negli indici a più chiavi • È stato corretto il piano di spiegazione e l'output del profiler per `$out` • È stato aggiunto un timeout per le operazioni dal sistema di monitoraggio interno per migliorare l'affidabilità • È stato corretto un difetto che influiva sui predicati delle query passati agli indici a più chiavi

Data	Note di rilascio
15 gennaio 2021	Nuove funzionalità Amazon DocumentDB 3.6 (versione della patch del motore 1.0.206295) • Possibilità di utilizzare un indice con la fase di aggregazione `$lookup` • `find()`le interrogazioni con proiezioni possono essere fornite a partire da un indice (interrogazione coperta) • Possibilità di utilizzo `hint()` con `findAndModify` • Ottimizzazioni delle prestazioni per l'operatore `$addToSet` • Miglioramenti per ridurre le dimensioni complessive dell'indice • Nuovi operatori di aggregazione: `$ifNull` `$replaceRoot` ,`$setIsSubset` ,, `$setInstersection` `$setUnion` , e `$setEquals` • Gli utenti possono anche terminare i propri cursori senza richiedere il ruolo `KillCursor`
9 novembre 2020	Nuove funzionalità Amazon DocumentDB 3.6 (versione della patch del motore 1.0.206295) • Nessuno Correzioni di bug e altre modifiche Amazon DocumentDB 3.6 (versione della patch del motore 1.0.206295) • Nessuno

Data	Note di rilascio
30 ottobre 2020	Nuove funzionalità Amazon DocumentDB 3.6 (versione della patch del motore 1.0.206295) • È stata aggiunta la possibilità di aprire un cursore del flusso di modifiche a livello di cluster o e database (`client.watch()` `mongo.watch()` `(db.watch())` • Possibilità di aumentare il periodo di conservazione del flusso di modifiche a 7 giorni (in precedenza 24 ore) Correzioni di bug e altre modifiche Amazon DocumentDB 3.6 (versione della patch del motore 1.0.206295) • Vari miglioramenti generali delle prestazioni • Un miglioramento mirato della sicurezza • È stato risolto un problema che impediva l'ordinamento nel secondo campo di un indice composto • Abilita l'indice regolare per l'uguaglianza su un singolo campo di un indice a più chiavi (non composto) • Condizione di gara di autenticazione fissa • È stato risolto un problema che causava un raro arresto anomalo della raccolta dei rifiuti • Miglioramento della sicurezza RBAC • Metrica aggiunta `databaseConnectionsMax` • Miglioramenti delle prestazioni per determinati carichi di lavoro sulle istanze `r5.24xlarge`

Data	Note di rilascio
22 settembre 2020	Nuove funzionalità Amazon DocumentDB 3.6 (versione della patch del motore 1.0.20629 5) • `$out`fase di aggregazione • È stato aumentato il numero massimo di connessioni e cursore per istanza fino a 10 volte Correzioni di bug e altre modifiche Amazon DocumentDB 3.6 (versione della patch del motore 1.0.20629 5) • Nessuno
10 luglio 2020	Nuove funzionalità Amazon DocumentDB 3.6 (versione della patch del motore 1.0.20629 5) • Copia istantanea interregionale Correzioni di bug e altre modifiche Amazon DocumentDB 3.6 (versione della patch del motore 1.0.20629 5) • Nessuno

Data	Note di rilascio
30 giugno 2020	Nuove funzionalità Amazon DocumentDB 3.6 (versione della patch del motore 1.0.206295) • Istanze medie T3 Correzioni di bug e altre modifiche Amazon DocumentDB 3.6 (versione della patch del motore 1.0.206295) • Recupero della memoria inattiva per le istanze t3 • Miglioramenti dell'autenticazione • Prestazioni di autenticazione SASL migliorate • È stato risolto `currentOp` il problema relativo al superamento del numero massimo di operazioni possibili • È stato risolto `killOps` il problema relativo all'aggiornamento e all'eliminazione in blocco • Miglioramenti alle `$sample` prestazioni con `$match` • Supporto fisso per la `$$` fase di redazione in cond case • Sono state risolte varie cause ricorrenti di crash • Miglioramenti allo sweeping TTL per ridurre iOS e la latenza • Utilizzo ottimizzato della memoria per `$unwind` • Risolto il problema della raccolta delle statistiche sulle condizioni di gara con indice di caduta • È stata corretta la condizione di gara durante la creazione simultanea dell'indice • Risolto un problema di crash occasionale nell'indice `hash_search`

Note di rilascio generali di Amazon DocumentDB

Aggiornamenti generali del servizio, tra cui disponibilità regionale, integrazioni di strumenti e funzionalità tra versioni diverse.

Data	Note di rilascio
15 settembre 2026	Nuova funzionalità Il tipo di istanza serverless di Amazon DocumentDB è ora disponibile nelle versioni del motore Amazon DocumentDB 5.0 e 8.0 nelle seguenti regioni: • Asia Pacifico (Osaka) • Asia Pacifico (Giacarta) • Asia Pacifico (Malesia) • Asia Pacifico (Melbourne) • Asia Pacifico (Thailandia) • Canada occidentale (Calgary) • Europa (Zurigo) • Europa (Stoccolma) • Israele (Tel Aviv) • Messico (centrale) Per ulteriori informazioni, consulta Classi di istanze supportate per regione e Utilizzo di Amazon DocumentDB serverless.
29 aprile 2026	Nuova funzionalità Amazon DocumentDB è ora disponibile nella regione Canada occidenta le (Calgary). Per ulteriori informazioni, consulta questo post del blog.
17 aprile 2026	Nuova funzionalità È ora disponibile una scalabilità più rapida su DocumentDB serverless.
11 febbraio 2026	Nuova funzionalità

Data	Note di rilascio
	Amazon DocumentDB è ora disponibile nella regione Asia Pacifico (Melbourne). Per ulteriori informazioni, consulta questo post del blog. Amazon DocumentDB è ora disponibile nella regione Europa (Zurigo). Per ulteriori informazioni, consulta questo post del blog.
8 gennaio 2026	Nuova funzionalità Amazon DocumentDB è ora disponibile nella regione Asia Pacifico (Jakarta). Per ulteriori informazioni, consulta questo post del blog.
13 ottobre 2025	Nuova funzionalità Amazon DocumentDB ora supporta le istanze R6G e T4G nella regione Asia Pacifico (Hyderabad). Per ulteriori informazioni, consulta Classi di istanze supportate per regione.
7 ottobre 2025	Nuova funzionalità Amazon DocumentDB è ora disponibile nelle seguenti regioni: • Asia Pacifico (Malesia) • Asia Pacifico (Osaka) • Asia Pacifico (Thailandia) • Messico (centrale) Per ulteriori informazioni, consulta questo post del blog.
26 settembre 2025	Nuove funzionalità Amazon DocumentDB ora supporta la copia e la condivisione di istantanee su più regioni e account. Per ulteriori informazioni, consulta Copiare istantanee tra Regioni AWS e Condivisione delle istantanee del cluster Amazon DocumentDB.

Data	Note di rilascio
18 giugno 2025	Nuova funzionalità Amazon DocumentDB è ora disponibile nella regione Israele (Tel Aviv).
8 maggio 2025	Nuova funzionalità Amazon DocumentDB è ora disponibile nella regione Europa (Stoccolma). Per ulteriori informazioni, consulta questo post del blog.
2 aprile 2025	Correzioni di bug e altre modifiche È stato corretto un bug nella creazione di indici vettoriali con indici in background.
6 febbraio 2025	Nuova funzionalità Amazon DocumentDB è ora integrato con. AWS Toolkit for Visual Studio Code Per ulteriori informazioni, consulta questo post del blog What's New e consulta Working with Amazon DocumentDB in the Toolkit nella Guida per l'AWS Toolkit for Visual Studio Code utente.
22 ottobre 2024	Nuova funzionalità Amazon DocumentDB ora supporta metriche `StorageNetworkReceiveThroughput` di throughput della rete di storage e. `StorageNetworkTransmitThroughput` `StorageNetworkThroughput` Per ulteriori informazioni, consulta Valutazione dell'utilizzo delle istanze Amazon DocumentDB con metriche CloudWatch.
18 settembre 2024	Nuova funzionalità Amazon DocumentDB è ora disponibile nella regione Africa (Città del Capo). Per ulteriori informazioni, consulta questo post del blog. Amazon DocumentDB è ora disponibile nella regione Europa (Spagna). Per ulteriori informazioni, consulta questo post del blog.

Data	Note di rilascio
25 giugno 2024	Nuova funzionalità L'autenticazione con AWS IAM ARNs è disponibile nei cluster 5.0 basati su istanze Amazon DocumentDB in tutte le regioni supportate. Per ulteriori informazioni, consulta Autenticazione tramite identità IAM.
3 aprile 2024	Nuova funzionalità Amazon DocumentDB è ora disponibile nella regione del Medio Oriente (Emirati Arabi Uniti). Per ulteriori informazioni, consulta questo post del blog. Correzioni di bug e altre modifiche • È stato corretto un errore durante l'invocazione `db.coll.stats()` su mongo shell versione 1.7 e successive. • È stato risolto un problema di perdita di memoria per le query del flusso di modifiche che contengono `$regex` come parte della stessa pipeline di aggregazione.
30 gennaio 2024	Nuove funzionalità Cluster globali Amazon DocumentDB I cluster globali sono ora disponibili in entrambe le AWS GovCloud (US) regioni: AWS GovCloud (US-East) e AWS GovCloud (). US-West
13 dicembre 2023	Nuove funzionalità È stato aggiunto il supporto per la connettività EC2 con 1 clic. Per ulteriori informazioni, consulta Connettiti tramite Amazon EC2.
17 novembre 2023	Nuove funzionalità Amazon DocumentDB è ora disponibile nella regione AWS GovCloud ()US-East. Per ulteriori informazioni, consulta questo post del blog.

Data	Note di rilascio
25 settembre 2023	Nuove funzionalità Amazon DocumentDB è ora disponibile nella regione Asia Pacifico (Hong Kong). Per ulteriori informazioni, consulta questo post del blog.
11 settembre 2023	Nuove funzionalità Amazon DocumentDB è ora disponibile nella regione Asia Pacifico (Hyderabad). Per ulteriori informazioni, consulta questo post del blog.
29 aprile 2022	Nuove funzionalità Amazon DocumentDB è ora disponibile nella regione Cina (Pechino). Per ulteriori informazioni, consulta questo post del blog.
16 marzo 2022	Nuove funzionalità Amazon DocumentDB è ora disponibile nella regione Europa (Milano). Per ulteriori informazioni, consulta questo post del blog.
8 febbraio 2022	Nuove funzionalità Le istanze Amazon DocumentDB R6G e T4G sono ora disponibili in Asia Pacifico, Sud America ed Europa. Per ulteriori informazioni, consulta questo post del blog.
04 maggio 2021	Nuove funzionalità Scopri tutte le nuove funzionalità in questo post del blog. https://aws.amazon.com/about-aws/whats-new/2021/05/amazon-documentdb-improves-mongodb-compatibility-and-indexing-improvements/
9 novembre 2020	Nuove funzionalità Scopri tutte le nuove funzionalità in questo post del blog.
30 ottobre 2020	Nuove funzionalità Scopri tutte le nuove funzionalità in questo post del blog.

Data	Note di rilascio
22 settembre 2020	Nuove funzionalità Scopri tutte le nuove funzionalità in questo post del blog.
10 luglio 2020	Nuove funzionalità Scopri tutte le nuove funzionalità in questo post del blog.
30 giugno 2020	Nuove funzionalità Scopri tutte le nuove funzionalità in questo post del blog.

Note sulla versione dei cluster elastici Amazon DocumentDB

Note sulla versione per i cluster elastici Amazon DocumentDB.

Data	Note di rilascio
28 gennaio 2025	Nuova funzionalità Per una connettività con un solo clic, Amazon DocumentDB è ora integrato con cluster elastici e basati su AWS CloudShell istanze. Per ulteriori informazioni, consulta una o tutte le seguenti opzioni: • Inizia a usare Amazon DocumentDB • Inizia a usare i cluster elastici Amazon DocumentDB • Cos'è il nuovo post del blog • Post tecnico su come fare sul blog
12 novembre 2024	Nuove funzionalità I cluster elastici Amazon DocumentDB ora supportano gli indici in background.
1 novembre 2024	Nuova funzionalità

Data	Note di rilascio
	Amazon DocumentDB ora supporta le azioni di manutenzione elastica dei cluster. Per ulteriori informazioni, consulta Manutenzione dei cluster elastici di Amazon DocumentDB.
8 agosto 2024	Nuova funzionalità I cluster elastici Amazon DocumentDB sono ora disponibili nelle regioni Asia Pacifico (Hong Kong), Canada (Centrale) ed Europa (Parigi). Per ulteriori informazioni, consulta Disponibilità delle regioni per i cluster elastici.
8 luglio 2024	Nuova funzionalità I cluster elastici Amazon DocumentDB sono ora disponibili nella regione Europa (Milano). Per ulteriori informazioni, consulta Disponibilità delle regioni per i cluster elastici.
22 febbraio 2024	Nuove funzionalità Cluster elastici Amazon DocumentDB I cluster elastici Amazon DocumentDB ora supportano le seguenti funzionalità: • Repliche leggibili di istanze shard secondarie: per ulteriori informazioni, consulta il passaggio 5b di. Fase 1: Creare un cluster elastico • Start/stop cluster: per ulteriori informazioni, vedere. Arresto e avvio di un cluster elastico Amazon DocumentDB • Istanze shard configurabili: per ulteriori informazioni, vedere il passaggio 5b di. Fase 1: Creare un cluster elastico • Backup automatici per istantanee: per ulteriori informazioni, consulta. Gestione di un backup automatico dello snapshot di un cluster elastico • Copia istantanea: per ulteriori informazioni, consulta. Copiare un'istantanea di un cluster elastico

Data	Note di rilascio
30 gennaio 2024	Nuove funzionalità Cluster elastici Amazon DocumentDB I cluster elastici Amazon DocumentDB sono ora disponibili nelle seguenti regioni: • Asia Pacifico (Mumbai) • Asia Pacifico (Seul) • Sud America (San Paolo) • Europa (Londra) Per ulteriori informazioni, consulta Disponibilità della regione e della versione di Elastic Cluster.

3 agosto 2023	Nuove funzionalità
	
	Cluster elastici Amazon DocumentDB
	
	• I cluster Amazon DocumentDB Elastic ora supportano le seguenti operazioni:
	  • `top`
	  • `collStats`
	  • `hint`
	  • `dataSize`
	
	  Consulta API, operazioni e tipi di dati MongoDB supportati in Amazon DocumentDB l'elenco completo dei comandi e delle operazioni supportati.
	• Gli indici Time to Live (TTL) sono ora supportati.
	• `hints`Gli indici sono ora supportati con le espressioni di indice.

Data	Note di rilascio
22 marzo 2023	Nuove funzionalità I cluster elastici Amazon DocumentDB sono ora disponibili nelle regioni Asia Pacifico (Singapore), Asia Pacifico (Sydney) e Asia Pacifico (Tokyo). Per ulteriori informazioni, consulta Disponibilità della regione e della versione di Elastic Cluster.
30 novembre 2022	Nuove funzionalità Cluster elastici Amazon DocumentDB I cluster elastici Amazon DocumentDB sono un nuovo tipo di cluster Amazon DocumentDB che consente agli utenti di sfruttare le API di sharding MongoDB per scalare il proprio cluster. I cluster elastici gestiscono praticamente qualsiasi numero di letture e scritture con petabyte di capacità di archiviazione distribuendo i dati e l'elaborazione su più istanze e volumi di calcolo sottostanti. Per ulteriori informazi oni, consulta Utilizzo dei cluster elastici Amazon DocumentDB. https://docs.aws.amazon.com/documentdb/latest/devguide/docdb-using-elastic-clusters

Cronologia dei documenti per la Amazon DocumentDB Developer Guide

- Versione API: 31-10-2014
- Ultimo aggiornamento della documentazione: 27 maggio 2026

La tabella seguente descrive la documentazione per questa versione della Amazon DocumentDB Developer Guide.

Modifica	Descrizione	Data
Nuova guida per sviluppatori di Amazon DocumentDB	La Amazon DocumentDB Developer Guide è ora pubblicata con un nuovo URL. L'API Reference è disponibile separatamente su Amazon DocumentDB API Reference.	27 maggio 2026
AWS aggiornamento della politica gestita - modifica della politica	Amazon DocumentDB aggiorna le politiche di accesso completo per i cluster elastici.	11 febbraio 2025
AWS aggiornamento della politica gestita - modifica della politica	Amazon DocumentDB aggiorna le politiche di accesso completo per i cluster elastici.	21 febbraio 2024
AWS aggiornamento gestito delle politiche - modifica delle politiche	Amazon DocumentDB aggiorna le politiche di sola lettura e di accesso completo per i cluster elastici.	21 giugno 2023
AWS aggiornamento gestito delle politiche: nuova politica	Amazon DocumentDB introduce una nuova policy	8 giugno 2023

	di sola lettura per i cluster elastici.	
AWS aggiornamento gestito delle politiche: nuova politica	Amazon DocumentDB introduce una nuova politica di accesso completo per i cluster elastici.	5 giugno 2023
Compatibilità con MongoDB 5.0	Amazon DocumentDB è ora compatibile con la versione 5.0 di MongoDB.	1 marzo 2023
Aggiornamento delle politiche	Per supportare la funzionalità del cluster elastico di Amazon DocumentDB, la AmazonDocDBConsoleFullAccess policy viene aggiornata e AmazonDocDB-ElasticServiceRolePolicy viene introdotta.	30 novembre 2022
Cluster elastici	È stata aggiunta una nuova funzionalità Elastic Cluster che supporta il partizionamento (sharding) basato su hash dei dati nel sistema di storage distribuito di Amazon DocumentDB.	30 novembre 2022
Cluster globali	È stata aggiunta la documentazione su come utilizzare Global Clusters.	2 giugno 2021
Abbonamenti agli eventi	È stata aggiunta la documentazione sulla sottoscrizione agli eventi.	26 marzo 2021

Aggiornamenti alla versione 3.6	Miglioramenti documentati alla versione 3.6 nei controlli di accesso basati sui ruoli, negli operatori di aggregazione e nelle prestazioni.	15 gennaio 2021
Compatibilità con MongoDB 4.0	Amazon DocumentDB è ora compatibile con la versione 4.0 di MongoDB.	9 novembre 2020
Guide introduttive	Nuove guide introduttive per iniziare a usare Amazon DocumentDB utilizzando Amazon EC2 AWS Cloud9, Robo3T o Studio3T.	15 agosto 2020
Zone di disponibilità aggiuntive supportate	Amazon DocumentDB ha aggiunto il supporto per un'ulteriore zona di disponibilità in Asia Pacifico (Seoul) (ap-northeast-2).	14 luglio 2020
È stato aggiunto il supporto per la copia di istantanee tra le regioni.	Amazon DocumentDB ha aggiunto il supporto per la copia di istantanee del cluster. Regioni AWS Per ulteriori informazioni, consulta Copiare le istantanee tra le regioni.	10 luglio 2020
È stato aggiunto il supporto per la classe di istanza T3.	È stato aggiunto il supporto per i tipi di istanze T3 in tutte le regioni che supportano Amazon DocumentDB. Per ulteriori informazioni, consulta Classi di istanze supportate per regione e Specifiche delle classi di istanza.	30 giugno 2020

È stato aggiunto il supporto per AWS GovCloud (US).	Amazon DocumentDB è ora disponibile nella AWS GovCloud (US) regione (us-gov-west-1).	29 giugno 2020
Sono state CloudWatch aggiunte 16 nuove metriche.	Amazon DocumentDB ha aggiunto il supporto per 16 nuove CloudWatch metriche Amazon. Per ulteriori informazioni, consulta Monitoraggio di Amazon DocumentDB con. CloudWatch	23 giugno 2020
È stato aggiunto il supporto per i caratteri nulli e l'operatore $regex.	Amazon DocumentDB ha aggiunto il supporto per i caratteri nulli nelle stringhe e la possibilità di utilizzare un indice per $regex. Per visualizzare le API MongoDB supportate e le funzionalità di pipeline di aggregazione per Amazon DocumentDB, consulta Functional Differences with MongoDB. https://docs.aws.amazon.com/documentdb/latest/devguide/functional-differences.html	22 giugno 2020

È stato aggiunto il supporto per funzionalità di indicizzazione multi-chiave migliorate.	Amazon DocumentDB ha aggiunto il supporto per funzionalità di indicizzazione multi-chiave migliorate che includono l'indicizzazione di array più grandi di 2.048 byte e la possibilità di creare un indice a più chiavi composto con più chiavi nello stesso array. Per ulteriori informazioni, consulta Differenze funzionali con MongoDB.	23 aprile 2020
È stato aggiunto il supporto per la protezione dall'eliminazione per uno stack Amazon DocumentDB. CloudFormation	Amazon DocumentDB ha aggiunto il supporto per abilitare la protezione dalle eliminazioni durante la creazione di uno stack Amazon CloudFormation DocumentDB.	20 aprile 2020
È stato aggiunto il supporto per il controllo degli accessi basato sui ruoli.	Amazon DocumentDB ha aggiunto il supporto per il controllo degli accessi basato sui ruoli utilizzando ruoli integrati.	26 marzo 2020
È stato aggiunto il supporto per una zona di disponibilità aggiuntiva in Canada (centrale) (ca-central-1).	Amazon DocumentDB è ora disponibile nella regione Canada (centrale) (ca-central-1) con istanze di classe R5 e 3 zone di disponibilità.	26 marzo 2020

È stato aggiunto il supporto per due API MongoDB aggiuntive.	Amazon DocumentDB ha aggiunto il supporto per `$dateFromString` e le API `executionStats` MongoDB.	23 marzo 2020
È stato aggiunto il supporto per cinque API MongoDB aggiuntive.	Amazon DocumentDB ha aggiunto il supporto per le `$arrayTo0 bject` API `$objectTo Array` `$slice`,`$mod`, e `$range` MongoDB.	06 febbraio 2020
È stato aggiunto il supporto per il Canada (Central).	Amazon DocumentDB è ora disponibile nella regione Canada (Centrale) (ca-centr al-1) con istanze di classe R5.	11 dicembre 2019
`ChangeStreamLogSize`È stato aggiunto il supporto per.	Amazon DocumentDB ha aggiunto il supporto `ChangeStreamLogSize` per le metriche di Cloudwatch.	22 novembre 2019
È stato aggiunto il supporto per la regione Europa (Parigi)	Amazon DocumentDB è ora disponibile nella regione Europa (Parigi) (eu-west-3) con istanze di classe R5.	30 ottobre 2019
È stato aggiunto il supporto per la regione Asia Pacifico (Mumbai)	Amazon DocumentDB è ora disponibile nella regione Asia Pacifico (Mumbai) (ap-south-1) con istanze di classe R5.	17 ottobre 2019
È stato aggiunto il supporto per tre API MongoDB aggiuntive	Amazon DocumentDB ha aggiunto il supporto per le API `$addFields` `$concatAr rays` , e MongoDB. `$lookup`	16 ottobre 2019

È stato aggiunto il supporto per la regione Asia Pacifico (Singapore)	Amazon DocumentDB è ora disponibile nella regione Asia Pacifico (Singapore) (ap-southeast-1) con istanze di classe R5.	14 ottobre 2019
È stato aggiunto un nuovo documento per l'aggiornamento dei certificati TLS	Sono state aggiunte istruzioni per l'aggiornamento dei certificati CA per utilizzare il nuovo certificato CA per creare connessioni TLS.	02 ottobre 2019
È stato aggiunto il supporto API per i certificati	Amazon DocumentDB è un nuovo tipo di dati di certificato per le istanze. Per ulteriori informazioni, consulta DBInstance.	1° ottobre 2019
Supporto per la profilazione delle query	Amazon DocumentDB ha aggiunto la possibilità di profilare le operazioni supportate sulle istanze e sui database del cluster.	19 agosto 2019
È stata aggiunta la terza AZ in Asia Pacifico (Tokyo)	Amazon DocumentDB ha aggiunto una terza zona di disponibilità (AZ) per le istanze di calcolo in Asia Pacifico (Tokyo).	9 agosto 2019

Supporto per API Mongo aggiuntive	È stato aggiunto il supporto per funzionalità aggiuntive della pipeline di aggregazione che includono gli operatori `$in`, `$isoWeek` `$isoWeekYear` `$isoDayOfWeek` , e di aggregazione e la fase di `$dateToString` aggregazione. `$addToSet` Amazon DocumentDB ha inoltre aggiunto il supporto per il `top()` comando per la diagnostica a livello di raccolta e la possibilità di modificare il `expireAfterSeconds` parametro per gli indici TTL utilizzando il comando. `collMod()`	31 luglio 2019
È stato aggiunto il supporto per l'Europa (Londra)	Amazon DocumentDB è ora disponibile in Europa (Londra) (eu-west-2) con istanze di classe R5.	18 luglio 2019
Sono stati aggiunti esempi di codice	Sono stati aggiunti esempi di codice in R e Ruby per la connessione programmatica ad Amazon DocumentDB.	17 luglio 2019
È stata aggiunta una best practice	È stata aggiunta una best practice per aiutarti a gestire i costi di Amazon DocumentDB.	17 luglio 2019

Supporto per l'arresto e l'avvio di un cluster	Amazon DocumentDB ha aggiunto il supporto per l'arresto e l'avvio dei cluster per gestire i costi degli ambienti di sviluppo e test.	1° luglio 2019
Supporto per la protezione dall'eliminazione dei cluster	Per proteggere i cluster dall'eliminazione accidentale, Amazon DocumentDB ha aggiunto la protezione dall'eliminazione. Per ulteriori informazioni, consulta i seguenti argomenti: Creazione di un cluster Amazon DocumentDB, Modifica di un cluster Amazon DocumentDB, Eliminazione di un cluster Amazon DocumentDB e nell'argomento API DBCluster. `DeletionProtection` https://docs.aws.amazon.com/documentdb/latest/APIReference/API_DBCluster.html	1° luglio 2019
Aggiornamento delle differenze funzionali	Aggiunte le transazioni implicite alle differenze funzionali.	26 giugno 2019
Aggiunta delle differenze funzionali	È stata aggiunta una nota relativa all'archiviazione e alla compressione degli indici in Amazon DocumentDB.	13 giugno 2019
Regione aggiuntiva supportata	Amazon DocumentDB è ora disponibile in Asia Pacifico (Sydney) (ap-southeast-2) con istanze di classe R5.	5 giugno 2019

Classe di istanza R5 supportata in regioni aggiuntive	È stato aggiunto il supporto per la classe di istanza R5 per 4 regioni aggiuntive: Stati Uniti orientali (Ohio), Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (Oregon) e UE (Irlanda). Con questa modifica, le istanze R5 sono supportate in tutte le regioni che supportano Amazon DocumentDB.	17 maggio 2019
Regioni aggiuntive supportate	È stato aggiunto il supporto per 2 regioni aggiuntive, Asia Pacifico (Tokyo) (ap-northeast-1) e Asia Pacifico (Seoul) (ap-northeast-2) con classi di istanze R5. Per ulteriori informazioni, consultare Classi di istanze supportate in base alle regioni e Specifiche delle classi delle istanze.	8 maggio 2019
Sono stati aggiunti altri esempi di codici di connessione	Sono stati aggiunti esempi di codice in Java e C# per la connessione ad Amazon DocumentDB.	24 aprile 2019

Supporto aggiuntivo per l'API Mongo	Aggiunto il supporto per sette operatori di stringa di aggregazione (`$indexOfBytes` , `$indexOfCP` , `$strLenBytes` , `$strLenCP` , `$toLower`, `$toUpper`, `$split`), nove operatori data-ora (`$dayOfYear` , `$dayOfMonth` , `$dayOfWeek` , `$year`, `$month`, `$hour`, `$minute`, `$second`, `$millisecond`) e la fase `$sample` della pipeline di aggregazione.	4 aprile 2019
Sono stati aggiunti esempi di codici di connessione	Sono stati aggiunti esempi di codice in Python Node.js, PHP e Go per la connessione ad Amazon DocumentDB.	21 marzo 2019
Supporto per la regione di Francoforte e le istanze R5	È stato aggiunto il supporto per la regione Europa (Francoforte) (eu-central-1) con classi di istanze R5. Per ulteriori informazioni, consultare Classi di istanze supportate in base alle regioni e Specifiche delle classi delle istanze.	13 marzo 2019

Supporto per gli operatori di pipeline di aggregazione	Aggiunto il supporto per i nuovi operatori di stringa di aggregazione (`$concat`, `$substr`, `$substrBytes` , `$substrCP` , `$strcasecmp`), un operatore aggregazione di array (`$size`), un operatore accumulatore per gruppo di aggregazione (`$push`) e fasi di aggregazione (`$redact` e `$indexStats`). Aggiunto anche il supporto per gli operatori di array posizionali (`$[]` e `$[<identifier>]`) e `hint()`.	28 febbraio 2019
Aggiornamenti del motore	Aggiunta la documentazione per determinare le modifiche del cluster in sospeso e per aggiornare la versione del motore del cluster.	15 febbraio 2019
Eventi di controllo	È stato aggiunto il supporto per il controllo degli eventi del database con Amazon CloudWatch Logs.	12 febbraio 2019
Quick Start	È stato aggiunto un argomento di avvio rapido per aiutarti a iniziare facilmente con Amazon DocumentDB utilizzando. CloudFormation	11 gennaio 2019

Rilascio pubblico	Questa è la versione pubblica iniziale di Amazon DocumentDB (con compatibilità con MongoDB). Questa release include la guida per gli sviluppatori e la documentazione integrata di riferimento sulle API per la gestione delle risorse.	9 gennaio 2019

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.