aws

Guía para desarrolladores

Amazon DocumentDB



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

Amazon DocumentDB: Guía para desarrolladores

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

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon, de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

Qué es Amazon DocumentDB 1
Descripción general de 1
Clústeres 3
instancias 4
Regiones y zonas de disponibilidad 8
Regions 8
Zonas de disponibilidad 8
Precios 11
Prueba gratuita 12
Supervisión 12
Interfaces 12
Consola de administración de AWS 12
AWS CLI 13
Controladores de MongoDB 13
Siguientes pasos 13
Funcionamiento 14
Versiones de Amazon DocumentDB 14
Puntos de conexión de Amazon DocumentDB 17
Compatibilidad con TLS 21
Almacenamiento de Amazon DocumentDB 21
Replicación de Amazon DocumentDB 22
Fiabilidad de Amazon DocumentDB 22
Opciones de preferencia de lectura 24
Eliminaciones de TTL 29
Recursos facturables 29
¿Qué es una base de datos de documentos? 32
Casos de uso 33
Descripción de documentos 34
Trabajo con documentos 40
Guía de introducción 52
Requisitos previos 52
Paso 1: creación de un clúster 54
Paso 2: Conéctese a su clúster 57
Paso 3: Insertar y consultar datos 58

Paso 4: Explorar ... 60
Inicio rápido a usar CloudFormation ... 60
Requisitos previos ... 61
Lanzamiento de una Amazon DocumentDB CloudFormation pila ... 62
Acceso al clúster de Amazon DocumentDB ... 67
Protección de terminación y protección de eliminación ... 68
Migración y actualización ... 69
Guía de inicio rápido ... 69
Preparación del origen de DMS ... 70
Configuración de DMS ... 70
Habilitación de la compresión de DocumentDB ... 70
Creación de una tarea de replicación ... 70
Monitorear el progreso ... 71
Información adicional ... 72
Manual de procedimientos de migración ... 72
Compatibilidad ... 73
Detección de la carga de trabajo ... 76
Cree un nuevo clúster de destino ... 79
Migración de índices ... 80
Migración de usuario ... 81
Migración de datos ... 83
Supervisión ... 94
Validación ... 96
Migración desde Couchbase Server ... 102
Introducción ... 102
Comparación con Amazon DocumentDB ... 103
Discovery ... 106
Planificación ... 111
Migración ... 114
Validación ... 123
Fechas de soporte de la versión del motor ... 126
Calendario de lanzamientos de las principales versiones de Amazon DocumentDB ... 127
Calendario de lanzamientos de las versiones secundarias de Amazon DocumentDB ... 127
Actualización de una versión principal del motor ... 128
Rutas de actualización admitidas ... 129
Requisitos previos ... 130

Prácticas recomendadas ... 132
Qué cambia tras la actualización a Amazon DocumentDB 8.0 ... 135
Post-upgrade consideraciones para los clústeres actualizados desde la versión 3.6 o 4.0 ... 136
Aplicación de la actualización ... 137
Resolución de problemas ... 139
Actualización de una versión secundaria del motor ... 140
Antes de realizar una actualización de versión secundaria ... 140
Actualización a una versión secundaria ... 141
Verificar la actualización de la versión secundaria ... 142
Actualización mediante AWS DMS ... 142
Paso 1: habilitación de los flujos de cambios ... 143
Paso 2: modificación de la duración de la retención de los flujos de cambios ... 143
Paso 3: migración de los índices ... 143
Paso 4: Cree un AWS DMS instancia de replicación ... 145
Paso 5: Cree un AWS DMS punto final de origen ... 146
Paso 6: Crear una AWS DMS punto final de destino ... 149
Paso 7: crear y ejecutar una tarea de migración ... 151
Paso 8: cambiar el punto de conexión de origen de la aplicación al clúster de Amazon DocumentDB de destino ... 153
Uso de una versión de soporte a largo plazo (LTS) ... 153
Versión actual de LTS ... 153
¿Quién debe usar LTS ... 154
¿Cómo crear un clúster LTS nuevo ... 154
Cómo actualizar de la versión 3.6.0 o 4.0.0 del motor a un clúster LTS 5.0.0 ... 155
¿Cómo comprobar si el clúster de la versión 5.0.0 de mi motor actual está en LTS ... 155
Soporte extendido ... 156
Información general del soporte extendido ... 157
Precios del Soporte extendido ... 157
Responsabilidades ... 158
Características y configuraciones ... 159
Estado del motor ... 159
Tipos de clústeres ... 159
Características ... 160
Tipos de instancias ... 162
AWS integraciones ... 162
Elección de regiones y zonas de disponibilidad ... 163

Disponibilidad por región ... 164
OpenSearch integración ... 167
Amazon OpenSearch Service como destino ... 167
Limitaciones ... 171
En desarrollo ... 172
Conexión a DocumentDB ... 172
Conexión mediante programación ... 173
Conexión desde fuera de una Amazon VPC ... 203
Conexión como conjunto de réplicas ... 205
Conexión mediante Studio 3T ... 210
Conéctese usando DataGrip ... 219
Conectarse mediante Amazon EC2 ... 224
Conexión mediante el controlador JDBC ... 262
Conexión mediante el controlador ODBC ... 278
Programación con Amazon DocumentDB ... 292
Guía de programación de DocumentDB en Java ... 292
Cómo utilizar la validación de esquemas JSON ... 325
Consultas ... 336
Consulta de documentos ... 336
Plan de consulta ... 339
Descripción de los resultados ... 341
Planificador de consultas v2 ... 346
Planificador de consultas v3 ... 362
Datos geoespaciales ... 367
Índice parcial ... 373
Búsqueda de texto ... 381
Índices ... 389
Administración de índices ... 389
Tipos de índices ... 401
Propiedades de índices ... 410
Uso de flujos de cambios ... 429
Operaciones admitidas ... 430
Eventos ampliados ... 430
Facturación ... 431
Limitaciones ... 431
Habilitación de flujos de cambios ... 432

Uso de flujos de cambios con Python 434
Búsqueda completa de documentos 437
Usa $change StreamSplitLargeEvent para dividir eventos grandes en fragmentos 438
Reanudación de un flujo de cambios 438
Reanudación con `startAtOperationTime` 440
Reanudación con `postBatchResumeToken` 442
Transacciones 443
Modificación de la duración de la retención de registros 443
Flujos de cambios en instancias secundarias 447
Transacciones 448
Requisitos 449
Prácticas recomendadas 449
Limitaciones 449
Supervisión y diagnóstico 450
Nivel de aislamiento de transacciones 451
Casos de uso 452
Comandos admitidos 490
Capacidades no compatibles 490
Sesiones 491
Errores de la transacción 493
Reintento de las escrituras 494
Requisitos 494
Operaciones admitidas 494
Limitaciones 495
Habilitación 496
Funcionamiento 496
Gestión de errores 496
Migración 497
Uso de la intercalación 497
Limitaciones 498
Uso de vistas del 498
Prácticas recomendadas 499
Compatibilidad con los operadores de agregadores 500
Aprovechando los índices y las vistas 500
IA generativa 500
SageMaker Lienzo 501

Búsqueda vectorial ... 504
Utilización AWS Lambda con flujos de cambio ... 518
Limitaciones ... 519
Compatibilidad con MongoDB ... 520
Compatibilidad con MongoDB 8.0 ... 520
Novedades de Amazon DocumentDB 8.0 ... 520
Introducción a Amazon DocumentDB ... 521
Actualice o migre a Amazon DocumentDB 5.0 u 8.0 ... 522
Diferencias funcionales ... 522
Compatibilidad con MongoDB 5.0 ... 522
Novedades de Amazon DocumentDB 5.0 ... 523
Novedades de Amazon DocumentDB 5.0.1 ... 523
Introducción a Amazon DocumentDB 5.0 ... 521
Actualice o migre a Amazon DocumentDB 5.0 ... 522
Diferencias funcionales ... 522
Compatibilidad con MongoDB 4.0 ... 527
Características de Amazon DocumentDB 4.0 ... 527
Introducción a Amazon DocumentDB 4.0 ... 528
Actualización o migración a Amazon DocumentDB 4.0 ... 529
Diferencias funcionales ... 529
Diferencias funcionales con MongoDB ... 530
Beneficios funcionales de Amazon DocumentDB ... 530
Diferencias funcionales actualizadas ... 532
Diferencias funcionales con MongoDB ... 535
API, operaciones y tipos de datos de MongoDB admitidos ... 544
Comandos de la base de datos ... 545
Operadores de consulta y proyección ... 555
Operadores de actualización ... 558
Geospatial (Geoespacial) ... 560
Métodos de cursor ... 562
Operadores de canalización de agregación ... 564
Tipos de datos ... 582
Índices y propiedades de índices ... 583
Operadores de canalización de agregación ... 585
Geospatial (Geoespacial) ... 1220
Operadores de consulta y proyección ... 1245

Operadores de actualización ... 1314
Administración de clústeres e instancias ... 1369
Información general sobre las tareas operativas ... 1369
Agregación de una réplica a un clúster de Amazon DocumentDB ... 1370
Descripción de clústeres e instancias ... 1371
Creación de una instantánea de un clúster ... 1373
Restauración a partir de una instantánea ... 1374
Eliminación de una instancia de un clúster ... 1375
Eliminación de un clúster ... 1376
Clústeres globales ... 1376
¿Qué es un clúster global? ... 1376
¿Para qué sirven los clústeres globales? ... 1377
¿Cuáles son las limitaciones actuales de los clústeres globales? ... 1377
Guía de inicio rápido ... 1378
Administración de clústeres globales ... 1390
Conexión a clústeres globales ... 1398
Supervisión de clústeres globales ... 1398
Recuperación ante desastres ... 1399
Administración de clústeres de ... 1425
Descripción de clúster ... 1426
Configuración del clúster ... 1429
Configuración de almacenamiento en clústeres ... 1432
Determinación del estado de un clúster ... 1435
Ciclo de vida del clúster ... 1437
Escalado de clústeres ... 1481
Clonación de un volumen de clúster ... 1484
Información general de tolerancia a errores para un clúster ... 1498
Administración de instancias ... 1499
Determinación del estado de una instancia ... 1499
Ciclo de vida de la instancia ... 1499
Administración de clases de instancias ... 1524
NVMe-backed instancias ... 1537
Administración de grupos de subredes ... 1540
Creación de un grupo de subredes ... 1541
Descripción de un grupo de subredes ... 1547
Modificación de un grupo de subredes ... 1550

Eliminación de un grupo de subredes ... 1553
Alta disponibilidad y replicación ... 1554
Escalado de lectura ... 1555
Alta disponibilidad ... 1555
Adición de réplicas de ... 1557
Conmutación por error ... 1557
Retraso de replicación ... 1562
Administración de eventos ... 1564
Visualización de categorías de eventos ... 1564
Visualización de eventos de Amazon DocumentDB ... 1567
Administración de grupos de parámetros de clúster ... 1570
Descripción de los grupos de parámetros de clúster ... 1570
Creación de grupos de parámetros de clúster ... 1578
Modificación de grupos de parámetros de clúster ... 1581
Modificación de clústeres para utilizar grupos de parámetros de clúster personalizados ... 1586
Copia de grupos de parámetros de clúster ... 1587
Restablecimiento de los grupos de parámetros de clúster ... 1590
Eliminación de grupos de parámetros de clúster ... 1593
Referencia de parámetros de clúster ... 1596
Descripción de puntos de conexión ... 1620
Búsqueda de puntos de conexión de un clúster ... 1621
Búsqueda del punto de conexión de una instancia ... 1623
Conexión a puntos de conexión ... 1627
Descripción de los ARN de Amazon DocumentDB ... 1628
Creación de un nombre ARN ... 1628
Búsqueda de un ARN ... 1632
Etiquetado de recursos ... 1635
Información general de las etiquetas de recursos de ... 1635
Restricciones de las etiquetas ... 1636
Adición o actualización de etiquetas ... 1637
Enumeración de etiquetas ... 1638
Eliminación de etiquetas ... 1640
Mantenimiento de Amazon DocumentDB ... 1641
Acciones de mantenimiento para Amazon DocumentDB ... 1643
Numeración de versiones ... 1643
Administración de los periodos de mantenimiento ... 1645

Notificaciones de parches del motor 1647
Vista de mantenimiento pendiente 1649
Actualizaciones del motor 1652
Versiones secundarias 1656
Actualizaciones del sistema operativo 1656
User-initiated actualizaciones 1661
Clústeres globales 1662
Descripción de los roles vinculados a servicios 1663
Service-linked permisos de rol 1663
Crear un rol vinculado a servicios 1665
Modificación de un rol vinculado a servicios 1666
Eliminar un rol vinculado a un servicio 1666
Regiones compatibles con los roles vinculados al servicio de Amazon DocumentDB 1667
Optimización del rendimiento 1668
Problemas de rendimiento comunes 1668
CPU elevada 1668
Problemas de conectividad 1671
La consulta se ejecuta lentamente 1678
Anti-patterns 1682
Índice compuesto antipatrón: más de 3 atributos 1683
Consultas de larga duración 1685
Índices poco utilizados y redundantes 1688
Multi-key índices con matrices grandes 1691
Consejos para mejorar el rendimiento 1692
1. Utilice $match como primera etapa en las canalizaciones de agregación 1692
2. Utilice $project en Aggregation Pipeline para minimizar el tamaño de los datos de la canalización 1693
3. Habilite la compresión de documentos para reducir los costes de almacenamiento y mejorar el rendimiento de las consultas I/O 1693
4. Aproveche los índices para obtener un rendimiento óptimo de las consultas 1694
5. Optimice los modelos de datos en función de los patrones de consulta 1695
Análisis del plan de consultas 1697
Plazo general 1699
Etapas de ejecución 1699
Comprensión de los documentos examinados en Explicar las estadísticas de ejecución del plan 1701

Index-only escaneos 1709
Cuando Amazon DocumentDB utiliza un escaneo que solo incluye índices 1710
Requisitos para una consulta cubierta 1712
Restricciones 1713
Confirmación de un análisis que solo incluye índices 1713
Prácticas recomendadas 1715
Planifique los filtros de caché 1715
Comandos admitidos 1716
Formas de consulta compatibles 1718
Configurar, enumerar y borrar filtros 1718
Comprobar que se ha aplicado un filtro 1720
Ejemplos 1721
Comportamiento con sugerencias y cambios en el índice 1724
Prácticas recomendadas 1724
Directrices operativas básicas 1725
Determinación del tamaño de las instancias 1726
Uso de índices 1728
Prácticas recomendadas de seguridad 1730
Optimización de costos 1730
Uso de métricas para identificar los problemas de rendimiento 1731
TTL y cargas de trabajo de series temporales 1735
Migraciones 1736
Uso de grupos de parámetros de clúster 1736
Consultas de canalización de agregación 1737
BatchInsert y BatchUpdate 1737
Administrar índices 1737
Creación de índices de Amazon DocumentDB 1737
Mantenimiento de índices de Amazon DocumentDB 1744
Administración de la compresión de documentos 1746
Administración de la compresión de documentos 1747
Supervisión de la compresión de documentos 1749
Administración de la compresión basada en diccionarios 1750
Consideraciones sobre el rendimiento 1751
Habilitar la compresión basada en diccionarios 1751
Introducción 1752
Supervisión 1752

Seguridad 1754
Administración de contraseñas con AWS Secrets Manager 1755
Limitaciones de la integración de Secrets Manager con Amazon DocumentDB 1756
Descripción general de la administración de las contraseñas de los usuarios principales con AWS Secrets Manager 1756
Imponer a Amazon DocumentDB la administración de la contraseña del usuario principal en AWS Secrets Manager 1757
Administración de la contraseña del usuario principal para un clúster con Secrets Manager 1757
Protección de datos 1758
Client-side cifrado a nivel de campo 1759
Cifrado de datos en reposo 1766
Cifrado de datos en tránsito 1774
Administración de claves 1785
Gestión de identidad y acceso 1786
Público 1786
Autenticación con identidades 1787
Administración del acceso con políticas 1788
Cómo funciona Amazon DocumentDB con IAM 1790
Identity-based ejemplos de políticas 1797
Resolución de problemas 1800
Administración de permisos de acceso para los recursos de Amazon DocumentDB 1802
Uso de políticas basadas en identidades (políticas de IAM) 1807
AWS políticas administradas para Amazon DocumentDB 1812
Referencia sobre los permisos de la API de Amazon DocumentDB 1831
Cross-service confusa prevención adjunta 1840
Autenticación mediante la identidad de IAM 1841
Introducción a los roles y usuarios de IAM 1842
Configuración de tipos de cómputo 1845
Supervisión de solicitudes de autenticación 1846
Uso de la autenticación de IAM 1847
Controladores compatibles con IAM 1847
Preguntas frecuentes sobre la autenticación de identidad de IAM 1847
Autenticación mediante SCRAM-SHA-256 1848
Creación de usuarios 1849
Actualización de usuarios 1850

Autenticación ... 1850
Administración de usuarios de Amazon DocumentDB ... 1851
Usuario principal y `administrador de servicio` ... 1851
Creación de usuarios adicionales ... 1852
Rotación automática de contraseñas ... 1854
Role-Based Control de acceso ... 1855
Conceptos de RBAC ... 1856
Introducción a los roles integrados de RBAC ... 1858
Introducción a los roles definidos por el usuario de RBAC ... 1861
Conexión a Amazon DocumentDB como un usuario ... 1866
Comandos comunes ... 1868
Diferencias funcionales ... 1873
Límites ... 1873
Acceso a la base de datos mediante el control de Role-Based acceso ... 1873
Actualización de certificados ... 1882
Actualización de la aplicación y del clúster de Amazon DocumentDB ... 1882
Rotación automática de certificados del servidor ... 1887
Resolución de problemas ... 1888
Preguntas frecuentes ... 1888
Validación de conformidad ... 1895
Resiliencia ... 1896
Seguridad de la infraestructura ... 1897
Puntos de conexión de VPC (AWS PrivateLink) ... 1898
Consideraciones para los puntos de conexión de VPC ... 1899
Disponibilidad por región ... 1900
Creación de un punto de conexión de VPC de interfaz para la API de Amazon DocumentDB ... 1901
Creación de una política de puntos de conexión de VPC para la API de Amazon DocumentDB ... 1901
Prácticas recomendadas de seguridad ... 1903
Auditoría de eventos ... 1904
Eventos admitidos ... 1905
Habilitar auditorías ... 1910
Deshabilitar auditorías ... 1918
Acceso a los eventos ... 1921
Filtrado de eventos de DML ... 1921

Amazon VPC y Amazon DocumentDB 1927
Clústeres de DocumentDB en una VPC 1927
Acceso a un clúster en una VPC 1940
Cree una IPv4-only VPC para un clúster 1943
Creación de una VPC de pila doble para un clúster 1950
Copia de seguridad y restauración 1961
Conceptos de copia de seguridad y restauración 1962
Descripción del uso de almacenamiento de copias de seguridad en 1965
Volcado, restauración, importación y exportación de datos 1967
mongodump 1967
mongorestore 1968
mongoexport 1969
importación de mongo 1969
Tutorial 1970
Consideraciones sobre las instantáneas de clústeres 1972
Almacenamiento de copia de seguridad 1973
Intervalo de copia de seguridad 1974
Backup retention period (Periodo de retención de copia de seguridad) 1976
Copia del cifrado de instantáneas de clúster 1976
Diferencias entre las instantáneas automáticas y manuales 1977
Creación de una instantánea manual del clúster 1979
Copia de una instantánea de clúster 1982
Copia de instantáneas compartidas 1983
Copiar instantáneas de un lado a otro Regiones de AWS 1983
Limitaciones 1983
Tratamiento del cifrado 1984
Consideraciones relativas al grupo de parámetros 1984
Copia de una instantánea de clúster 1985
Compartir una instantánea de un clúster 1995
Cómo compartir una instantánea cifrada 1995
Uso compartido de una instantánea 1999
Restauración de una instantánea del clúster 2002
Restauración a un momento dado 2010
Eliminación de una instantánea del clúster 2016
Supervisión y registro 2018
Monitorización del estado de un clúster 2019

Valores de estado del clúster ... 2020
Monitorización del estado de un clúster ... 2022
Monitorización del estado de una instancia ... 2023
Valores de estado de instancia ... 2024
Supervise el estado de la instancia mediante el Consola de administración de AWS o AWS CLI ... 2027
Estado de una instancia ... 2029
Supervisar el estado de la instancia mediante el Consola de administración de AWS ... 2029
Visualización de recomendaciones de Amazon DocumentDB ... 2031
Suscripciones de eventos ... 2034
Suscripción a eventos ... 2035
Administración de las suscripciones de ... 2037
Categorías y mensajes ... 2041
Supervisión de Amazon DocumentDB con CloudWatch ... 2044
Métricas de Amazon DocumentDB ... 2045
Visualización CloudWatch de datos ... 2065
Dimensiones de Amazon DocumentDB ... 2072
Supervisión de las métricas de opcounter ... 2072
Supervisión de conexiones a bases de datos ... 2072
Registro de llamadas a la API de Amazon DocumentDB con CloudTrail ... 2073
Información de Amazon DocumentDB en CloudTrail ... 2073
Creación de perfiles de operaciones ... 2074
Operaciones admitidas ... 2075
Limitaciones ... 2076
Habilitación del generador de perfiles ... 2076
Deshabilitación del generador de perfiles ... 2081
Deshabilitación de la exportación de registros del generador de perfiles ... 2082
Acceso a los registros del generador de perfiles ... 2084
Consultas comunes ... 2085
Supervisión con información sobre rendimiento ... 2085
Conceptos de Información sobre rendimiento ... 2087
Activación y desactivación de Información sobre rendimiento ... 2091
Configuración de directivas de acceso para información sobre rendimiento ... 2094
Análisis de métricas mediante el panel de Información sobre rendimiento ... 2098
Recuperación de métricas con la API de Información sobre rendimiento ... 2119
CloudWatch Métricas de Amazon para Performance Insights ... 2134

Métricas de contador para Información sobre rendimiento 2137
Sin servidor 2139
Casos de uso de la tecnología sin servidor 2139
Uso de Amazon DocumentDB sin servidor para las cargas de trabajo aprovisionadas existentes 2141
Ventajas de Amazon DocumentDB sin servidor 2142
Cómo funciona sin servidor 2143
Descripción general de 2143
Configuraciones para clústeres de Amazon DocumentDB 2145
Capacidad de escalado de Amazon DocumentDB sin servidor 2145
Escalado de Amazon DocumentDB sin servidor 2147
Estado inactivo (0,5 DCU) 2149
Requisitos y limitaciones sin servidor 2150
Requisitos de características 2150
Creación de un clúster que utilice la tecnología sin servidor 2151
Creación de un clúster de Amazon DocumentDB sin servidor 2151
Añadido de una instancia de Amazon DocumentDB sin servidor 2154
Migración a sistemas sin servidor 2155
Migración de los clústeres de DocumentDB existentes a sistemas sin servidor 2155
Migración de MongoDB a DocumentDB sin servidor 2161
Administración sin sistemas sin servidor 2161
Visualización y modificación de la configuración del rango de capacidad de escalado de un clúster 2161
Visualización y modificación del nivel de promoción de los lectores sin servidor 2164
Límites de instancias sin servidor 2167
Configuración de escalado sin servidor 2169
Selección del rango de capacidad de escalado para un clúster de DocumentDB sin servidor 2169
Elección de la configuración `MinCapacity` para un clúster de DocumentDB sin servidor . 2170
Elección de la configuración `MaxCapacity` para un clúster de DocumentDB sin servidor . 2172
Evitar errores de memoria insuficiente 2174
¿Por qué no se reduce verticalmente mi instancia sin servidor? 2175
Supervisión sin servidor 2176
Memoria insuficiente: estado de parámetros incompatibles 2177
CloudWatch Métricas de Amazon para DocumentDB sin servidor 2177

Supervisión del rendimiento de DocumentDB sin servidor con Información sobre rendimiento ... 2182
Clústeres elásticos ... 2183
Casos de uso de clústeres elásticos ... 2184
Perfiles de usuario ... 2184
Administración de contenido y registros históricos ... 2184
Ventajas de los clústeres elásticos ... 2184
AWS integración de servicios ... 2184
Disponibilidad en regiones y versiones ... 2185
Disponibilidad regional para clústeres elásticos ... 2185
Disponibilidad de versiones ... 2186
Limitaciones ... 2186
Administración de clústeres elásticos ... 2186
Operaciones de consulta y escritura ... 2187
Gestión de colecciones e índices ... 2187
Administración y diagnóstico ... 2188
Opt-in features ... 2188
Cómo funcionan ... 2188
Partición en los clústeres elásticos de Amazon DocumentDB ... 2189
Migración de clústeres elásticos ... 2192
Escalado de clústeres elásticos ... 2192
Fiabilidad de los clústeres elásticos ... 2192
Almacenamiento y disponibilidad de clústeres elásticos ... 2193
Diferencias funcionales entre Amazon DocumentDB 4.0 y los clústeres elásticos ... 2193
Introducción ... 2195
Requisitos previos ... 2195
Paso 1: crear un clúster elástico ... 2197
Paso 2: conectarse a su clúster elástico ... 2203
Paso 3: hacer una partición de su colección; insertar y consultar los datos ... 2204
Paso 4: Explorar ... 2208
Prácticas recomendadas ... 2208
Elección de las claves de partición ... 2208
Administración de conexiones ... 2209
Colecciones sin particiones ... 2209
Escalado de clústeres elásticos ... 2209
Monitoreo de clústeres elásticos ... 2210

Administración de los clústeres elásticos ... 2211
Modificación de configuraciones de clústeres elásticos ... 2211
Monitoreo de un clúster elástico ... 2215
Eliminación de un clúster elástico ... 2219
Administración de instantáneas de los clústeres elásticos ... 2221
Detención e inicio de un clúster elástico ... 2236
Mantenimiento de clústeres elásticos ... 2241
Cifrado de datos en reposo ... 2249
Cómo utilizan los clústeres elásticos de Amazon DocumentDB las subvenciones en AWS KMS ... 2251
Creación de una clave administrada por el cliente ... 2251
Supervisión de las claves de cifrado de clústeres elásticos de Amazon DocumentDB ... 2253
Más información ... 2258
Service-linked roles ... 2259
Service-linked permisos de rol para clústeres elásticos ... 2259
Cuotas ... 2263
Tipos de instancias admitidas ... 2263
Regiones admitidas ... 2265
Cuotas regionales ... 2267
Cuotas de agregación ... 2270
Cuotas de clúster ... 2270
Cuotas de instancias ... 2272
Restricciones en la nomenclatura ... 2279
Restricciones de TTL ... 2281
Cuotas de clúster elásticas ... 2282
Cuotas de particiones de clústeres elásticos ... 2283
Cuotas de CPU, memoria, conexión y cursor de los clústeres elásticos por fragmento ... 2283
Resolución de problemas ... 2285
Problemas de conectividad ... 2285
No se puede conectar a un punto de conexión de Amazon DocumentDB ... 2285
Comprobación de una conexión a una instancia de Amazon DocumentDB ... 2291
Conexión a un punto de conexión no válido ... 2291
La configuración del controlador afecta al número de conexiones ... 2292
Índices ... 2292
Error al crear un índice ... 2293
El índice en segundo plano genera problemas de latencia y falla ... 2293

Sobrecarga del índice 2294
Rendimiento y utilización de recursos 2295
Consulta las estadísticas de inserción, actualización y eliminación 2296
Análisis del rendimiento de la memoria caché 2297
Busque y finalice las consultas bloqueadas o de larga duración 2299
Consulte un plan de consultas y optimice una consulta 2300
¿Cómo puedo ver un plan de consultas en clústeres elásticos? 2302
Enumera todas las operaciones en ejecución en una instancia 2305
Sepa cuándo avanza una consulta 2307
Determina por qué un sistema de repente funciona con lentitud 2310
Determinar la causa del uso elevado de la CPU 2312
Búsqueda de los cursores abiertos en una instancia 2313
Visualización de la versión actual del motor Amazon DocumentDB 2313
Análisis del uso de los índices e identificar los índices no utilizados 2314
Identificación de los índices omitidos 2316
¿Cómo determino la sobrecarga de colecciones de bases de datos? 2318
Resumen de consultas útiles 2319
Recopilación de elementos no utilizados 2320
Descripción de la recolección de basura en Amazon DocumentDB 2321
Proceso de recopilación de elementos no utilizados 2321
Arquitectura de almacenamiento y almacenamiento extendido 2322
Supervisión de recopilación de elementos no utilizados 2323
Ejemplo de salida de CollStats 2326
Preguntas frecuentes 2328
Códigos de error 2331
Notas de la versión 2365
Notas de lanzamiento de la versión del motor Amazon DocumentDB 2366
Notas generales de la versión de Amazon DocumentDB 2431
Notas de la versión de Amazon DocumentDB Elastic Clusters 2436
Historial de revisión 2441
..... mmcdliv

Qué es Amazon DocumentDB (compatible con MongoDB)

Amazon DocumentDB (compatible con MongoDB) es un servicio rápido, de confianza y completamente administrado. Amazon DocumentDB facilita la configuración, el funcionamiento y el MongoDB-compatible escalado de bases de datos en la nube. Con Amazon DocumentDB, puede ejecutar el mismo código de aplicación y utilizar los mismos controladores y herramientas que utiliza con MongoDB.

Antes de utilizar Amazon DocumentDB, debe revisar los conceptos y las características que se describen en Funcionamiento. A continuación, realice los pasos que se indican en Guía de introducción.

Temas

- Información general de Amazon DocumentDB
- Clústeres
- instancias
- Regiones y zonas de disponibilidad
- Precios de Amazon DocumentDB
- Supervisión
- Interfaces
- Siguientes pasos
- Funcionamiento de Amazon DocumentDB
- ¿Qué es una base de datos de documentos?

Información general de Amazon DocumentDB

A continuación, se muestran algunas de las características generales de Amazon DocumentDB:

- Amazon DocumentDB admite dos tipos de clústeres: clústeres basados en instancias y clústeres elásticos. Los clústeres elásticos admiten cargas de trabajo con una capacidad de reads/writes almacenamiento de millones por segundo y petabytes. Para obtener más información acerca de los clústeres elásticos, consulte Uso de clústeres elásticos de Amazon DocumentDB. El siguiente contenido hace referencia a los clústeres basados en instancias de Amazon DocumentDB.

- Amazon DocumentDB aumenta automáticamente el tamaño del volumen de almacenamiento a medida que aumentan las necesidades de almacenamiento de la base de datos. Su volumen de almacenamiento crece en incrementos de 10 GB, hasta un máximo de 256 TiB para la versión 8.0 del motor y posteriores (128 TiB para las versiones anteriores del motor). No necesita aprovisionar almacenamiento excesivo para el clúster a fin de afrontar el crecimiento en el futuro.
- Con Amazon DocumentDB, puede aumentar el rendimiento de lectura para admitir solicitudes de aplicaciones de gran volumen mediante la creación de hasta 15 instancias de réplica. Las réplicas de Amazon DocumentDB comparten el mismo almacenamiento subyacente, lo que reduce los costos y evita la necesidad de realizar escrituras en los nodos de réplica. Esta capacidad libera más potencia de procesamiento para atender las solicitudes de lectura y reduce el tiempo de retraso de la réplica, que a menudo se reduce a milisegundos de un solo dígito. Puede añadir réplicas en cuestión de minutos, independientemente del tamaño del volumen de almacenamiento. Amazon DocumentDB también proporciona un punto de conexión de lectura para que la aplicación pueda conectarse sin tener que realizar un seguimiento de las réplicas a medida que se añaden o quitan.
- Amazon DocumentDB le permite escalar los recursos informáticos y memoria para cada una de las instancias. Las operaciones de escalado de los recursos informáticos normalmente se llevan a cabo en cuestión de minutos.
- Amazon DocumentDB se ejecuta en Amazon Virtual Private Cloud (Amazon VPC), de modo que puede aislar su base de datos en su propia red virtual. También puede configurar ajustes del firewall para controlar el acceso de red al clúster.
- Amazon DocumentDB monitoriza continuamente el estado del clúster. Si se produce un error en una instancia, Amazon DocumentDB reinicia automáticamente la instancia y los procesos asociados. Amazon DocumentDB no requiere una reproducción de los reregistros de la base de datos durante la recuperación tras un bloqueo, lo que reduce considerablemente los tiempos de reinicio. Amazon DocumentDB también aísla la caché de la base de datos del proceso de la base de datos, lo que permite que la caché sobreviva a un reinicio de la instancia.
- Cuando ocurre un error en una instancia, Amazon DocumentDB automatiza la conmutación por error en una de las 15 réplicas de Amazon DocumentDB que crea en otras zonas de disponibilidad. Si no se han aprovisionado réplicas y se produce un error, Amazon DocumentDB intenta crear una nueva instancia de Amazon DocumentDB de forma automática.
- La capacidad de copia de seguridad de Amazon DocumentDB permite la recuperación a un momento dado del clúster. Esta característica le permite restaurar el clúster a cualquier segundo dentro de su período de retención, hasta los últimos 5 minutos. Puede configurar el período de retención de copia de seguridad automático hasta un máximo de 35 días. Las copias de seguridad

automatizadas se almacenan en Amazon Simple Storage Service (Amazon S3), diseñado para una durabilidad del 99,999999999 %. Las copias de seguridad de Amazon DocumentDB son automáticas, incrementales y continuas, y no afectan al rendimiento del clúster.

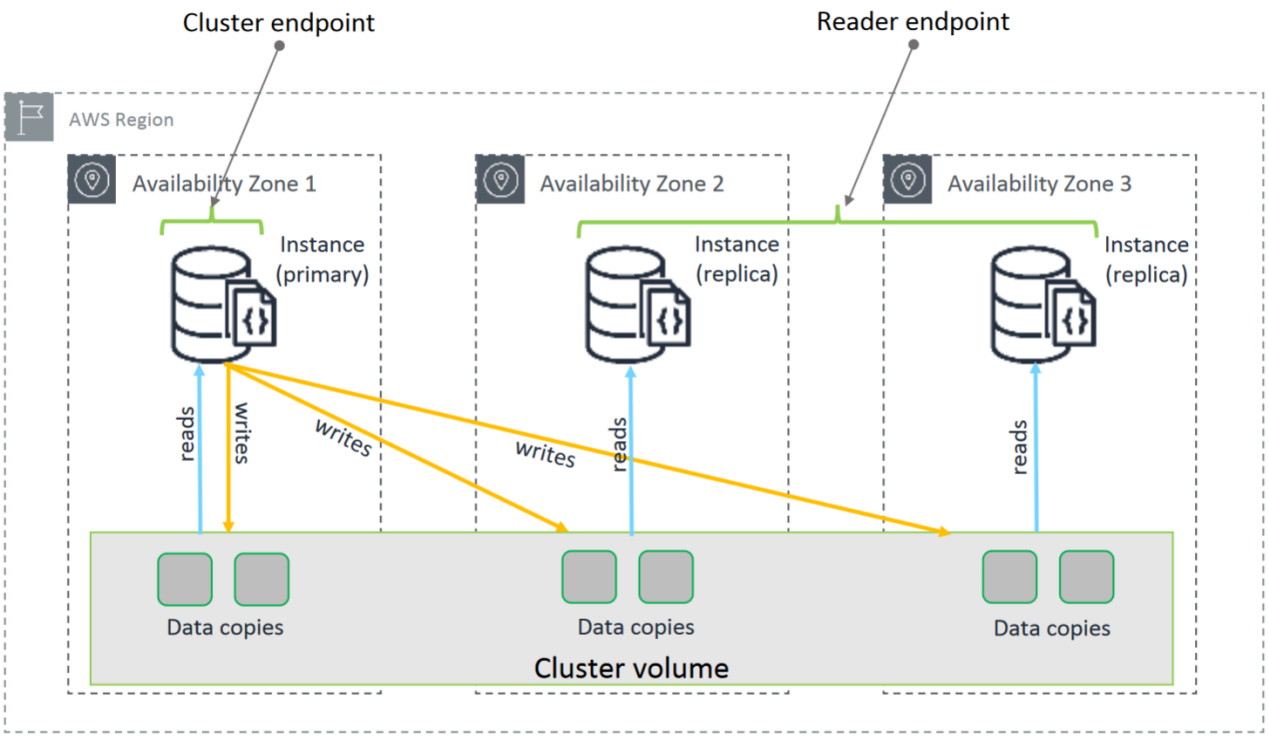
- Con Amazon DocumentDB, puede cifrar sus bases de datos mediante claves que puede crear y controlar mediante (). AWS Key Management Service AWS KMS En un clúster de base de datos que se ejecute con el cifrado de Amazon DocumentDB, los datos almacenados en reposo en el almacenamiento subyacente están cifrados. Las copias de seguridad automatizadas, las instantáneas y las réplicas que se encuentran en el mismo clúster también están cifradas.
- Amazon DocumentDB está autorizada por el Programa federal de administración de riesgos y autorizaciones (FedRAMP). Cuenta con la autorización FedRAMP High para las regiones AWS GovCloud (EE. UU.) y la autorización FedRAMP moderada para las regiones de EE. UU. AWS East/West Para obtener más información sobre los esfuerzos de cumplimiento AWS y las iniciativas de cumplimiento, consulte los AWS servicios según el programa de cumplimiento.

Si es la primera vez que utilizas AWS los servicios, usa los siguientes recursos para obtener más información:

- AWS ofrece servicios de informática, bases de datos, almacenamiento, análisis y otras funciones. Para obtener una descripción general de todos los AWS servicios, consulte Computación en la nube con Amazon Web Services.
- AWS proporciona varios servicios de bases de datos. Para averiguar cuál es el mejor servicio para su entorno, consulte Bases de datos en AWS.

Clústeres

Un clúster contiene entre 0 y 16 instancias y un volumen de almacenamiento del clúster que administra los datos de esas instancias. Todos los procesos de escritura se efectúan a través de la instancia principal. Todas las instancias (principales y réplicas) admiten operaciones de lectura. Los datos del clúster se almacenan en el volumen del clúster con copias en tres zonas de disponibilidad diferentes.



Los clústeres basados en instancias de Amazon DocumentDB 5.0 admiten dos configuraciones de almacenamiento para un clúster de bases de datos: Amazon DocumentDB estándar y Amazon DocumentDB. I/O-optimized Para obtener más información, consulte Configuraciones de almacenamiento en clústeres de Amazon DocumentDB.

instancias

Una instancia de Amazon DocumentDB es un entorno de base de datos aislado en la nube. Una instancia puede contener varias bases de datos creadas por el usuario. Puede crear y modificar una instancia mediante el o el. Consola de administración de AWS AWS CLI

La capacidad de computación y de memoria de una instancia se determina mediante su clase de instancia. Puede seleccionar la instancia que mejor se adapte a sus necesidades. Si sus necesidades cambian con el tiempo, puede elegir otra clase de instancia. Para ver las especificaciones de las clases de instancias, consulte Especificaciones de clases de instancias.

Las instancias de Amazon DocumentDB se ejecutan únicamente en el entorno de Amazon VPC. Amazon VPC le ofrece control sobre el entorno de red virtual: puede seleccionar un intervalo de direcciones IP propio, crear subredes y configurar el direccionamiento y las listas de control de acceso (ACL).

Antes de crear instancias de Amazon DocumentDB, debe crear un clúster que contenga las instancias.

No todas las clases de instancias son compatibles con todas Región de AWS. En la siguiente tabla, se muestran las clases de instancias compatibles en cada región.

> **Note**
>
> Para ver una lista completa de tipos de instancias compatibles con Amazon DocumentDB en cada clase de instancias, consulte Especificaciones de clases de instancias.

Clases de instancias admitidas por región

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Sin servidor
Este de EE. UU. (Ohio)	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	compatible
Este de EE. UU. (Norte de Virginia)	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	compatible
Oeste de EE. UU. (Oregón)	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	compatible
África (Ciudad del Cabo)			Soportado	Soportado		Soportado	Soportado	compatible
América del Sur (São Paulo)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Hong Kong)	Soportado		Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Hyderabad)			Soportado	Soportado		Soportado	Soportado	compatible

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Sin servidor
Asia-Pacífico (Malasia)			Soportado			Soportado	Soportado	compatible
Asia-Pacífico (Mumbai)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Osaka)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Seúl)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Sídney)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Yakarta)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Melbourne)			Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Singapur)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Tailandia)			Soportado			Soportado	Soportado	compatible
Asia-Pacífico (Tokio)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Canadá (centro)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Oeste de Canadá (Calgary)			Soportado	Soportado		Soportado	Soportado	compatible

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Sin servidor
Europa (Fráncfort)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Europa (Zúrich)		Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Europa (Irlanda)	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	compatible
Europa (Londres)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Europa (Milán)			Soportado	Soportado		Soportado	Soportado	compatible
Europa (París)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Europa (España)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Europa (Estocolmo)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
México (centro)			Soportado			Soportado	Soportado	compatible
Medio Oriente (EAU)			Soportado	Soportado		Soportado	Soportado	compatible
China (Pekín)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
China (Ningxia)	Soportado		Soportado	Soportado		Soportado	Soportado	compatible
Israel (Tel Aviv)			Soportado	Soportado		Soportado	Soportado	Soportado

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Sin servidor
AWS GovCloud (US-West)	Soportado	Soportado	Soportado	Soportado			Soportado	Soportado
AWS GovCloud (US-East)		Soportado	Soportado	Soportado		Soportado	Soportado	compatible

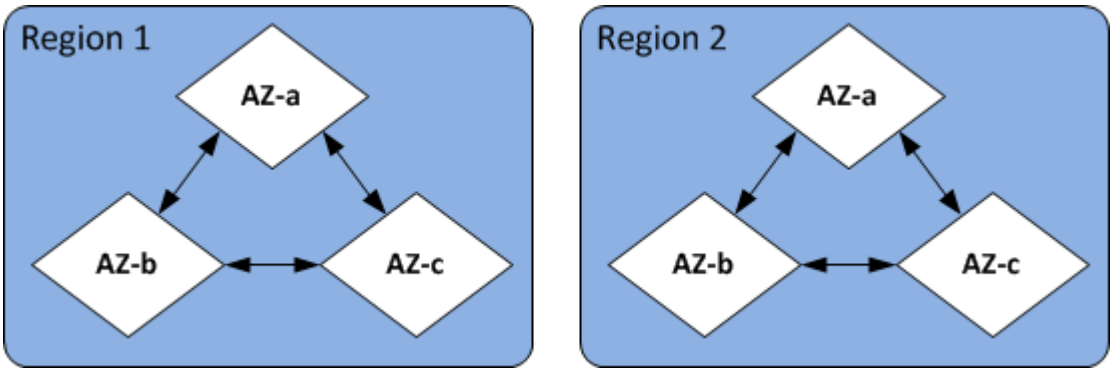
Regiones y zonas de disponibilidad

Regiones de AWS y las zonas de disponibilidad definen las ubicaciones físicas del clúster y las instancias.

Regions

AWS Los recursos de computación en la nube se alojan en centros de datos de alta disponibilidad en diferentes partes del mundo (por ejemplo, Norteamérica, Europa o Asia). Las ubicaciones del centro de datos se denominan regiones Región de AWS.

Cada uno Región de AWS está diseñado para estar completamente aislado del otro Regiones de AWS. Dentro de cada una de ellas hay varias zonas de disponibilidad. Al lanzar los nodos en zonas de disponibilidad diferentes, puede lograr la máxima tolerancia a errores. El siguiente diagrama muestra una vista de alto nivel del funcionamiento Regiones de AWS de las zonas de disponibilidad.



Zonas de disponibilidad

Cada una Región de AWS contiene varias ubicaciones distintas denominadas zonas de disponibilidad. Cada zona de disponibilidad está diseñada para estar aislada de los errores que se produzcan en otras zonas de disponibilidad y para proporcionar conectividad de red de baja

latencia económica con otras zonas de disponibilidad de la misma región. Al lanzar instancias para un clúster determinado en varias zonas de disponibilidad, puede proteger las aplicaciones en el caso improbable de que se produzca un error en una zona de disponibilidad.

La arquitectura de Amazon DocumentDB separa el almacenamiento y la computación. Para la capa de almacenamiento, Amazon DocumentDB replica seis copias de sus datos en tres zonas de AWS disponibilidad. Por ejemplo, si lanza un clúster de Amazon DocumentDB en una región que solo admite dos zonas de disponibilidad, el almacenamiento de datos se replicará de seis maneras en tres zonas de disponibilidad, pero las instancias de computación solo estarán disponibles en dos zonas de disponibilidad.

La siguiente tabla muestra el número de zonas de disponibilidad que puede usar en una determinada zona Región de AWS para aprovisionar instancias informáticas para su clúster.

Nombre de la región	Region	Zonas de disponibilidad (cálculo)
Este de EE. UU. (Ohio)	`us-east-2`	3
Este de EE. UU. (Norte de Virginia)	`us-east-1`	6
Oeste de EE. UU. (Oregón)	`us-west-2`	4
África (Ciudad del Cabo)	`af-south-1`	3
América del Sur (São Paulo)	`sa-east-1`	3
Asia-Pacífico (Hong Kong)	`ap-east-1`	3
Asia-Pacífico (Hyderabad)	`ap-south-2`	3
Asia-Pacífico (Malasia)	`ap-southeast-5`	3

Nombre de la región	Region	Zonas de disponibilidad (cálculo)
Asia-Pacífico (Mumbai)	ap-south-1	3
Asia-Pacífico (Osaka)	ap-northeast-3	3
Asia-Pacífico (Seúl)	ap-northeast-2	4
Asia-Pacífico (Singapur)	ap-southeast-1	3
Asia-Pacífico (Sídney)	ap-southeast-2	3
Asia-Pacífico (Yakarta)	ap-southeast-3	3
Asia-Pacífico (Melbourne)	ap-southeast-4	3
Asia-Pacífico (Tailandia)	ap-southeast-7	3
Asia-Pacífico (Tokio)	ap-northeast-1	3
Canadá (centro)	ca-central-1	3
Oeste de Canadá (Calgary)	ca-west-1	3
Región China (Pekín)	cn-north-1	3
China (Ningxia)	cn-northwest-1	3
Europa (Fráncfort)	eu-central-1	3
Europa (Zúrich)	eu-central-2	3
Europa (Irlanda)	eu-west-1	3
Europa (Londres)	eu-west-2	3

Nombre de la región	Region	Zonas de disponibilidad (cálculo)
Europa (Milán)	`eu-south-1`	3
Europa (París)	`eu-west-3`	3
Europa (España)	`eu-south-2`	3
Europa (Estocolmo)	`eu-north-1`	3
México (centro)	`mx-central-1`	3
Medio Oriente (EAU)	`me-central-1`	3
Israel (Tel Aviv)	`il-central-1`	3
AWS GovCloud (US-West)	`us-gov-west-1`	3
AWS GovCloud (US-East)	`us-gov-east-1`	3

Precios de Amazon DocumentDB

Los clústeres de Amazon DocumentDB se facturan en función de los siguientes componentes:

- Horas de instancia (por hora): en función de la clase de instancia (por ejemplo,`db.r5.xlarge`). Los precios se muestran por hora, pero las facturas se ajustan hasta el segundo y muestran las horas en formato decimal. El uso de Amazon DocumentDB se factura por incrementos de un segundo, con un mínimo de 10 minutos. Para obtener más información, consulte Administración de clases de instancias.
- I/O solicitudes (por cada millón de solicitudes al mes): número total de I/O solicitudes de almacenamiento que realiza en un ciclo de facturación.
- Almacenamiento de copias de seguridad (por GiB al mes): el almacenamiento de copias de seguridad es el almacenamiento asociado a copias de seguridad de base de datos automatizadas que haya realizado. Aumentar el período de retención de copia de seguridad u obtener instantáneas de base de datos adicionales aumenta el almacenamiento de copias de seguridad consumido por su base de datos. El almacenamiento de respaldo se mide GB-months y no se

aplica por segundo. Para obtener más información, consulte Copia de seguridad y restauración en Amazon DocumentDB.

- Transferencia de datos (por GB): transferencia de datos entrantes y salientes de la instancia desde o hacia Internet u otro Regiones de AWS medio.

Para obtener información detallada, consulte los precios de Amazon DocumentDB.

Prueba gratuita

Puede probar Amazon DocumentDB de forma gratuita con la versión de prueba gratuita de 1 mes. Para obtener más información, consulte la versión de prueba gratuita en el sitio de precios de Amazon DocumentDB o consulte las preguntas frecuentes sobre la prueba gratuita de Amazon DocumentDB.

Supervisión

Hay varias formas de hacer un seguimiento del rendimiento y el estado de una instancia. Puede utilizar el CloudWatch servicio gratuito de Amazon para supervisar el rendimiento y el estado de una instancia. Puede encontrar gráficos de rendimiento en la consola de Amazon DocumentDB. Puede suscribirse a eventos de Amazon DocumentDB si desea recibir una notificación cuando se produzcan cambios en una instancia, una instantánea, un grupo de parámetros o un grupo de seguridad.

Para obtener más información, consulte los siguientes temas:

- Supervisión de Amazon DocumentDB con CloudWatch
- Registro de llamadas a la API de Amazon DocumentDB con AWS CloudTrail

Interfaces

Hay varias formas de interactuar con Amazon DocumentDB, incluidas la Consola de administración de AWS y la AWS CLI.

Consola de administración de AWS

Consola de administración de AWS Se trata de una sencilla interfaz de usuario basada en la web. Desde la consola puede administrar sus clústeres e instancias sin necesidad de programación. Para

acceder a la consola de Amazon DocumentDB, inicie sesión en Consola de administración de AWS y abra la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb

AWS CLI

Puede usar el AWS Command Line Interface (AWS CLI) para administrar los clústeres e instancias de Amazon DocumentDB. Con una configuración mínima, puede comenzar a utilizar toda la funcionalidad que ofrece la consola de Amazon DocumentDB con el programa de terminal que desee.

- Para instalarlo AWS CLI, consulte Instalación de la interfaz de línea de AWS comandos.
- Para empezar a utilizarla AWS CLI para Amazon DocumentDB, consulte la referencia de la interfaz de línea de AWS comandos de Amazon DocumentDB.

Controladores de MongoDB

Para desarrollar y escribir aplicaciones en un clúster de Amazon DocumentDB, también puede utilizar los controladores de MongoDB con Amazon DocumentDB. Para obtener más información, consulte la pestaña del intérprete de comandos de MongoDB en Conexión con TLS habilitado o Conexión con TLS deshabilitado.

Siguientes pasos

En la sección anterior se han presentado los componentes de la infraestructura básica ofrecidos por Amazon DocumentDB. ¿Qué debería hacer a continuación? Dependiendo de sus circunstancias, consulte uno de los siguientes temas para empezar.

- Para empezar a usar Amazon DocumentDB, cree un clúster y una instancia utilizando. CloudFormation Inicio rápido de Amazon DocumentDB a usar CloudFormation
- Empiece a trabajar con Amazon DocumentDB creando un clúster y una instancia con las instrucciones que se proporcionan en nuestro Guía de introducción.
- Comience a utilizar Amazon DocumentDB creando un clúster elástico siguiendo las instrucciones que se indican en Introducción a los clústeres elásticos de Amazon DocumentDB.
- Migre su implementación de MongoDB a Amazon DocumentDB utilizando las instrucciones en Migración y actualización de Amazon DocumentDB

Funcionamiento de Amazon DocumentDB

Amazon DocumentDB (compatible con MongoDB) es un servicio de bases de datos totalmente administrado. MongoDB-compatible Con Amazon DocumentDB, puede ejecutar el mismo código de aplicación y utilizar los mismos controladores y herramientas que utiliza con MongoDB. Amazon DocumentDB es compatible con MongoDB 3.6, 4.0, 5.0 y 8.0.

Temas

- Versiones de Amazon DocumentDB
- Puntos de conexión de Amazon DocumentDB
- Compatibilidad con TLS
- Almacenamiento de Amazon DocumentDB
- Replicación de Amazon DocumentDB
- Fiabilidad de Amazon DocumentDB
- Opciones de preferencia de lectura
- Eliminaciones de TTL
- Recursos facturables

Versiones de Amazon DocumentDB

> **Note**
>
> La numeración de versiones secundarias se aplica a Amazon DocumentDB 5.0 y versiones posteriores. Amazon DocumentDB 3.6 y 4.0 solo utilizan versiones principales con parches de motor.

A partir de Amazon DocumentDB 5.0, la versión del motor utiliza un esquema de numeración de cuatro niveles: major.major.minor.patch.

- Versión principal: se identifica mediante los dos primeros números de la versión (por ejemplo, 5.0). La versión principal representa el nivel de compatibilidad de MongoDB. Las versiones principales introducen nuevas funciones, mejoras de rendimiento y actualizaciones de compatibilidad con MongoDB. La actualización entre versiones principales (por ejemplo, de la 4.0 a la 5.0)

requiere una actualización de la versión principal (MVU). Para obtener más información, consulte Actualización local de la versión principal Amazon DocumentDB.

- Versión secundaria: se identifica por el tercer número de la versión (por ejemplo, el 1 número 5.0.1). Las versiones secundarias incluyen mejoras como nuevas funciones, correcciones de seguridad y correcciones de errores dentro de la misma versión principal. Puede actualizar manualmente a una nueva versión secundaria. Para obtener más información, consulte Actualización de la versión secundaria de Amazon DocumentDB.
- Versión de parche: los parches del motor se aplican en una versión secundaria e incluyen correcciones de seguridad y errores críticos. La versión del parche se controla mediante la versión del parche del motor (por ejemplo,`3.0.17983`), que puede comprobar `db.runCommand({getEngineVersion: 1})` ejecutándola. Los parches se aplican durante el período de mantenimiento del clúster. Para obtener más información, consulte Mantenimiento de Amazon DocumentDB.

Al crear un clúster nuevo, puede especificar cualquier versión de motor disponible. Para obtener información sobre las versiones disponibles y las fechas de soporte, consulteFechas de soporte de la versión del motor de Amazon DocumentDB.

Cuando utilice Amazon DocumentDB, empezará creando un clúster. Un clúster se compone de cero o varias instancias y de un volumen de clúster que administra los datos de esas instancias. Un volumen de clúster de Amazon DocumentDB es un volumen de almacenamiento de base de datos virtual que abarca varias zonas de disponibilidad. Cada zona de disponibilidad tiene una copia de los datos del clúster.

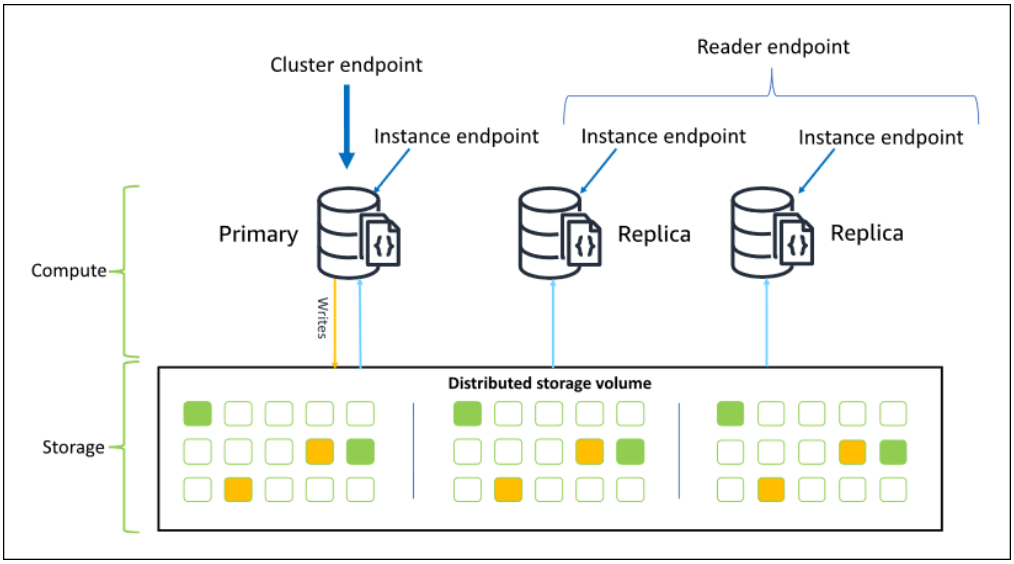
Los clústeres de Amazon DocumentDB constan de dos componentes principales:

- Volumen de clúster: utiliza un servicio de almacenamiento nativo en la nube para replicar los datos de seis maneras en tres zonas de disponibilidad, lo que proporciona un almacenamiento disponible y duradero. Un clúster de Amazon DocumentDB tiene exactamente un volumen de clúster, que puede almacenar hasta 256 TiB de datos para la versión 8.0 del motor y posteriores (128 TiB para las versiones anteriores del motor).
- Instancias: proporcionan la potencia de procesamiento de la base de datos al escribir y leer datos desde el volumen de almacenamiento del clúster. Un clúster de Amazon DocumentDB puede tener de 0 a 16 instancias.

Las instancias adoptan uno de estos dos roles:

- Instancia de base de datos principal: admite operaciones de lectura y escritura y realiza todas las modificaciones de los datos en el volumen de clúster. Cada clúster de Amazon DocumentDB tiene una instancia principal.
- Instancia de réplica: solo admite operaciones de lectura. Un clúster de Amazon DocumentDB puede tener hasta 15 réplicas, además de la instancia principal. El hecho de disponer de varias réplicas le permite distribuir las cargas de trabajo de lectura. Además, al colocar las réplicas en distintas zonas de disponibilidad, también puede aumentar la disponibilidad del clúster.

En el siguiente diagrama se ilustra la relación entre el volumen de clúster, la instancia principal y las réplicas de un clúster de Amazon DocumentDB:



Las instancias del clúster no tienen por qué ser de la misma clase, y se pueden aprovisionar y terminar según sea necesario. Esta arquitectura le permite escalar la capacidad de cómputo de su clúster de forma independiente de su almacenamiento.

Cuando la aplicación escribe datos en la instancia principal, la instancia principal ejecuta una operación de escritura duradera en el volumen del clúster. A continuación, replica el estado de esa escritura (no los datos) en cada réplica activa. Las réplicas de Amazon DocumentDB no participan en el procesamiento de escrituras y, por lo tanto, las réplicas de Amazon DocumentDB son ventajosas para el escalado de lectura. Las lecturas de réplicas de Amazon DocumentDB son, en última instancia, consistentes con un retardo de réplica mínimo, normalmente menos de 100 milisegundos una vez que la instancia principal escribe los datos. Se garantiza que las operaciones de lectura de las réplicas se realizan en el orden en que fueron escritas en la instancia principal. El retardo de la réplica varía en función de la frecuencia de cambio de datos, y los periodos con mucha actividad

de escritura pueden aumentar el retardo de la réplica. Para obtener más información, consulte las métricas `ReplicationLag` en Métricas de Amazon DocumentDB.

Puntos de conexión de Amazon DocumentDB

Amazon DocumentDB ofrece varias opciones de conexión para proporcionar una amplia variedad de casos de uso. Para conectarse a una instancia de un clúster de Amazon DocumentDB, debe especificar el punto de conexión de la instancia. Un punto de conexión es una dirección y un número de puerto de host, separados por dos puntos.

Conéctese a su clúster mediante el punto de enlace del clúster en el modo de conjunto de réplicas (consulte). Conexión a Amazon DocumentDB como conjunto de réplicas Usa este enfoque a menos que debas conectarte al punto final del lector o al punto final de la instancia. Para dirigir solicitudes a las réplicas, elija una configuración de preferencia de lectura de controlador que maximice el escalado de lectura mientras cumple con los requisitos de coherencia de lectura de su aplicación. Las preferencias de lectura `secondaryPreferred` permiten las lecturas de réplica y libera la instancia principal para hacer más trabajo.

Los siguientes puntos de conexión están disponibles en un clúster de Amazon DocumentDB.

Punto de conexión de clúster

El punto de conexión del clúster le conecta a la instancia principal del clúster. El punto de conexión del clúster se puede utilizar para operaciones de lectura y escritura. Un clúster de Amazon DocumentDB tiene exactamente un punto de conexión de clúster.

El punto de conexión del clúster proporciona conmutación por error para conexiones de lectura y escritura con el clúster. Si la instancia principal actual del clúster produce un error y el clúster tiene al menos una réplica de lectura activa, el punto de conexión del clúster redirige automáticamente las solicitudes de conexión a una nueva instancia principal. Cuando se conecte a su clúster de Amazon DocumentDB, utilice el punto de enlace del clúster en el modo de conjunto de réplicas (consulteConexión a Amazon DocumentDB como conjunto de réplicas).

A continuación, se muestra un punto de conexión de un clúster de Amazon DocumentDB de ejemplo:

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

A continuación, se muestra una cadena de conexión de ejemplo que usa este punto de conexión de clúster:

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

Para obtener información acerca de cómo encontrar los puntos de conexión de un clúster, consulte Búsqueda de puntos de conexión de un clúster.

Punto de conexión del lector

El punto de conexión del lector balancea la carga de las conexiones de solo lectura entre todas las réplicas disponibles del clúster. Un punto de conexión de lector de clúster funcionará como punto de conexión del clúster si se conecta a través del modo `replicaSet`, es decir, en la cadena de conexión, el parámetro del conjunto de réplicas es `&replicaSet=rs0`. En este caso, podrá llevar a cabo operaciones de escritura en el principal. Sin embargo, si se conecta al clúster que especifica `directConnection=true` y se intenta llevar a cabo una operación de escritura a través de una conexión con el punto de conexión de lector, se produce un error. Un clúster de Amazon DocumentDB tiene exactamente un punto de conexión de lector.

Si el clúster contiene solo una instancia principal, el punto de conexión del lector se conecta a la instancia principal. Cuando se añade una instancia de réplica al clúster de Amazon DocumentDB, el punto de conexión del lector abre las conexiones de solo lectura con la nueva réplica una vez que esté activo.

A continuación, se muestra un punto de conexión del lector de ejemplo de un clúster de Amazon DocumentDB:

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

A continuación, se muestra una cadena de conexión de ejemplo que usa un punto de conexión de lector:

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

El punto de conexión del lector balancea la carga de las conexiones de solo lectura, no de las solicitudes de lectura. Si algunas de las conexiones del punto de conexión del lector se utilizan más que otras, es posible que las solicitudes de lectura no se puedan equilibrar uniformemente entre las instancias del clúster. Se recomienda distribuir solicitudes mediante la conexión al punto de

conexión de clúster como un conjunto de réplicas y mediante la opción de preferencia de lectura secondaryPreferred.

Para obtener información acerca de cómo encontrar los puntos de conexión de un clúster, consulte Búsqueda de puntos de conexión de un clúster.

Punto de conexión de instancia

Un punto de conexión de una instancia se conecta a una instancia específica del clúster. El punto de conexión de instancia de la instancia principal actual se puede utilizar para realizar operaciones de lectura y escritura. Sin embargo, si se intentan realizar operaciones de escritura en un punto de conexión de instancia de una réplica de lectura, se produce un error. Un clúster de Amazon DocumentDB tiene un punto de conexión de instancia para cada instancia activa.

Un punto de conexión de instancia proporciona un control directo sobre las conexiones con una instancia específica en los casos en los que el uso del punto de conexión del clúster o del lector puede no ser adecuado. Un ejemplo de caso de uso es el aprovisionamiento de una carga de trabajo de análisis diarios de solo lectura. Puede aprovisionar una instancia de réplica mayor de lo normal, conectarse directamente a la nueva instancia más grande con su punto de conexión de instancia, ejecutar las consultas de análisis y después terminar la instancia. El uso del punto de conexión de instancia impide que el tráfico de los análisis afecte a otras instancias del clúster.

A continuación, se muestra un punto de conexión de instancia de ejemplo para una sola instancia de un clúster de Amazon DocumentDB:

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

A continuación, se muestra una cadena de conexión de ejemplo que usa este punto de conexión de instancia:

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

> **Note**
>
> El rol de instancia principal o réplica de una instancia puede cambiar debido a un evento de conmutación por error. Las aplicaciones no deben presuponer en ningún momento que un determinado punto de conexión de instancia es la instancia principal. No se conecte

a los puntos de enlace de la instancia para las aplicaciones de producción. En su lugar, usa el punto final del clúster en el modo de conjunto de réplicas (consultaConexión a Amazon DocumentDB como conjunto de réplicas). Para obtener información sobre el control avanzado de la prioridad de conmutación por error de las instancias, consulte Información general de la tolerancia a errores de clúster de Amazon DocumentDB.

Para obtener información acerca de cómo encontrar los puntos de conexión de un clúster, consulte Búsqueda del punto de conexión de una instancia.

Modo de conjunto de réplicas

Puede conectarse al punto de conexión de su clúster de Amazon DocumentDB en modo de conjunto de réplicas especificando el nombre del conjunto de réplicas `rs0`. La conexión en el modo de conjunto de réplicas proporciona la capacidad de especificar las opciones Read Concern, Write Concern y Read Preference. Para obtener más información, consulte Coherencia de lectura.

A continuación, se muestra una cadena de conexión de ejemplo que se conecta en el modo de conjunto de réplicas:

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

Cuando se conecta en modo de conjunto de réplicas, el clúster de Amazon DocumentDB se muestra ante los controladores y clientes como un conjunto de réplicas. Las instancias añadidas y eliminadas del clúster de Amazon DocumentDB se reflejan automáticamente en la configuración del conjunto de réplicas.

Cada clúster de Amazon DocumentDB consta de un único conjunto de réplicas con el nombre predeterminado `rs0`. El nombre del conjunto de réplicas no se puede modificar.

La conexión al punto de conexión del clúster en el modo de conjunto de réplicas es el método recomendado para uso general.

> **Note**
>
> Todas las instancias de un clúster de Amazon DocumentDB atienden las conexiones en el mismo puerto TCP.

Compatibilidad con TLS

Para obtener más información sobre cómo conectarse a Amazon DocumentDB mediante seguridad de la capa de transporte (TLS), consulte Cifrado de datos en tránsito.

Almacenamiento de Amazon DocumentDB

Los datos de Amazon DocumentDB se almacenan en el volumen del clúster, que es un volumen único y virtual que usa unidades de estado sólido (SSD). Un volumen de clúster se compone de seis copias de los datos, que se replican automáticamente en varias zonas de disponibilidad de una sola Región de AWS. Esta replicación ayuda a garantizar que los datos se conserven durante mucho tiempo, con menos riesgo de que se pierdan los datos. También ayuda a garantizar que el clúster esté más disponible durante una conmutación por error, porque ya existen copias de sus datos en otras zonas de disponibilidad. Estas copias pueden seguir enviando solicitudes de datos a las instancias del clúster de Amazon DocumentDB.

Cómo se factura el almacenamiento de datos de

Amazon DocumentDB aumenta automáticamente el tamaño del volumen del clúster cuando aumenta la cantidad de datos. Un volumen de clúster de Amazon DocumentDB puede crecer hasta un tamaño máximo de 256 TiB para la versión 8.0 del motor y posteriores (128 TiB para las versiones anteriores del motor). Sin embargo, solo se le cobrará por el espacio que utilice en un volumen de clúster de Amazon DocumentDB. A partir de Amazon DocumentDB 4.0, cuando se eliminan datos, por ejemplo, al eliminar una colección o un índice, el espacio asignado general disminuye en una cantidad comparable. Por lo tanto, puede reducir los cargos de almacenamiento eliminando tablas, índices y bases de datos que ya no necesite. En la versión 3.6 de Amazon DocumentDB, el volumen del clúster puede reutilizar el espacio liberado al eliminar datos, pero el volumen en sí nunca disminuye de tamaño. Como resultado, en la versión 3.6, es posible que no vea ningún cambio en el almacenamiento cuando elimine una colección o un índice, aunque el espacio liberado se vuelva a utilizar.

> **Note**
>
> Con Amazon DocumentDB 3.6, los costos de almacenamiento se basan en el “límite máximo” de almacenamiento (la cantidad máxima que se asignó al clúster de Amazon DocumentDB en cualquier momento). Puede administrar los costos evitando las prácticas de ETL que crean grandes volúmenes de información temporal o que cargan grandes volúmenes de datos nuevos antes de eliminar datos antiguos innecesarios. Si la eliminación

de datos de un clúster de Amazon DocumentDB produce una cantidad sustancial de espacio asignado pero que no se utiliza, el restablecimiento del nivel máximo de crecimiento exige realizar el volcado de datos lógicos y restablecerlo a un clúster nuevo, con una herramienta como `mongodump` o `mongorestore`. La creación y restauración de una instantánea no reduce el almacenamiento asignado debido a que la distribución física del almacenamiento subyacente no se verá modificada en la instantánea restaurada.

> **Note**
>
> El uso de utilidades `mongorestore` como `mongodump` y I/O conlleva cargos en función del tamaño de los datos que se leen y escriben en el volumen de almacenamiento.

Para obtener información sobre el almacenamiento de datos y los precios de Amazon DocumentDB, consulte las preguntas frecuentes sobre I/O precios y precios de Amazon DocumentDB (compatible con MongoDB). https://aws.amazon.com/documentdb/faqs/#Pricing

Replicación de Amazon DocumentDB

En un clúster de Amazon DocumentDB, cada instancia de réplica expone un punto de conexión independiente. Los puntos de conexión de estas réplicas proporcionan acceso de solo lectura a los datos del volumen de clúster. Le permiten escalar la carga de trabajo de lectura de los datos entre varias instancias replicadas. También ayudan a mejorar el rendimiento de las lecturas de datos y a aumentar la disponibilidad de los datos en el clúster de Amazon DocumentDB. Las réplicas de Amazon DocumentDB también son objetivos de conmutación por error y se promocionan rápidamente si se produce un error en la instancia principal del clúster de Amazon DocumentDB.

Fiabilidad de Amazon DocumentDB

Amazon DocumentDB está diseñado para ofrecer fiabilidad, durabilidad y tolerancia a errores. (Para mejorar la disponibilidad, debe configurar el clúster de Amazon DocumentDB de modo que tenga varias instancias de réplica en distintas zonas de disponibilidad). Amazon DocumentDB también incluye varias características automáticas que la convierten en una solución de base de datos de confianza.

Reparación automática del almacenamiento

Amazon DocumentDB mantiene varias copias de los datos en tres zonas de disponibilidad, lo que reduce considerablemente la posibilidad de pérdida de datos debido a un error de almacenamiento. Amazon DocumentDB también detecta automáticamente los errores del volumen de clúster. Cuando se produce un error en un segmento de un volumen del clúster, Amazon DocumentDB lo repara inmediatamente. Utiliza los datos de los demás volúmenes que conforman el volumen del clúster para ayudar a garantizar que los datos del segmento reparado estén actualizados. Por consiguiente, Amazon DocumentDB evita la pérdida de datos y reduce la necesidad de realizar una restauración a un momento dado para recuperarse del error de una instancia.

Calentamiento de caché que puede sobrevivir

Amazon DocumentDB administra su caché de páginas en un proceso independiente de la base de datos para que pueda sobrevivir independientemente de la base de datos. En el improbable caso de que se produzca un error de la base de datos, la caché de páginas permanece en la memoria. De este modo, se garantiza que el grupo del búfer contiene datos con el estado más actualizado al reiniciar la base de datos.

Recuperación de bloqueos

Amazon DocumentDB se ha diseñado para recuperarse de un bloqueo casi instantáneamente y continuar sirviendo sus datos de aplicación sin el registro binario. Amazon DocumentDB realiza las recuperaciones de bloqueos de forma asíncrona en subprocesos paralelos, de forma que su base de datos permanece abierta y disponible inmediatamente después de un bloqueo.

Administración de recursos

Amazon DocumentDB protege los recursos para los procesos críticos, como las comprobaciones de estado. Cuando una instancia experimenta una gran presión de memoria, Amazon DocumentDB limita las solicitudes. Es posible que algunas operaciones queden en cola hasta que disminuya la presión de la memoria. Si la presión de la memoria continúa, es posible que se agote el tiempo de espera de las operaciones en cola.

Controle la limitación de memoria con las siguientes CloudWatch métricas:`LowMemThrottleQueueDepth`,,,. `LowMemThrottleMaxQueueDepth` `LowMemNumOperationsThrottled` `LowMemNumOperationsTimedOut` Para obtener más información, consulte Supervisión de Amazon DocumentDB con CloudWatch.

Si observa una presión de memoria sostenida, amplíe la instancia para proporcionar memoria adicional para su carga de trabajo.

Opciones de preferencia de lectura

Amazon DocumentDB utiliza un servicio de almacenamiento compartido nativo en la nube que replica los datos seis veces en tres zonas de disponibilidad para ofrecer altos niveles de durabilidad. Amazon DocumentDB no se basa en la replicación de datos en varias instancias para lograr durabilidad. Los datos del clúster serán duraderos tanto si este contiene una sola instancia como si tiene 15.

Temas

- Durabilidad de escritura
- Aislamiento de lectura
- Coherencia de lectura
- Alta disponibilidad
- Escalado de operaciones de lectura

Durabilidad de escritura

Amazon DocumentDB utiliza un sistema de almacenamiento único, distribuido, tolerante a errores y de recuperación automática. Este sistema replica seis copias (V=6) de sus datos en tres zonas de AWS disponibilidad para proporcionar una alta disponibilidad y durabilidad. Al escribir datos, Amazon DocumentDB garantiza que todas las escrituras se registren de forma duradera en la gran mayoría de los nodos antes de confirmar la escritura al cliente. Si ejecuta un conjunto de réplicas de MongoDB de tres nodos y utiliza la opción Write Concern de `{w:3, j:true}`, obtendría la mejor configuración posible en comparación con Amazon DocumentDB.

Las operaciones de escritura en el clúster de Amazon DocumentDB debe procesarlas la instancia principal del clúster. Si se intenta escribir a una réplica, se producirá un error. Una operación de escritura confirmada desde una instancia principal de Amazon DocumentDB es duradera y no se puede revertir. Amazon DocumentDB es muy duradero de forma predeterminada y no admite una opción de escritura no duradera. No se puede modificar el nivel de durabilidad (es decir, Write Concern). Amazon DocumentDB ignora w=anything y, de hecho, es w: 3 y j: true. No puede reducirlo.

Debido a que en la arquitectura de Amazon DocumentDB la computación y el almacenamiento están separados, un clúster con una sola instancia es de larga duración. La durabilidad se gestiona en

la capa de almacenamiento. Como resultado, un clúster de Amazon DocumentDB con una sola instancia y uno con tres instancias consiguen el mismo nivel de durabilidad. Puede configurar el clúster para que se adapte a su caso de uso específico a la vez que proporciona un alto nivel de durabilidad para sus datos.

Las operaciones de escritura en un clúster de Amazon DocumentDB son atómicas dentro del mismo documento.

Amazon DocumentDB no admite la opción `wtimeout` y no devolverá un error si se especifica un valor. Las operaciones de escritura en la instancia principal de Amazon DocumentDB están aseguradas contra el bloqueo indefinido.

Aislamiento de lectura

Las operaciones de lectura de una instancia de Amazon DocumentDB solo devuelven datos que se conservan antes de que se inicie la consulta. Las operaciones de lectura nunca devuelven datos modificados una vez que se inicia la ejecución de la consulta, y las lecturas sucias no son posibles en ninguna circunstancia.

Coherencia de lectura

Los datos leídos de un clúster de Amazon DocumentDB son duraderos y no se pueden revertir. Puede modificar la lectura consistente para las operaciones de lectura de Amazon DocumentDB especificando la preferencia de lectura de la solicitud o conexión. Amazon DocumentDB no admite una opción de lectura no duradera.

Las lecturas desde una instancia principal del clúster de Amazon DocumentDB tienen una consistencia alta en condiciones funcionamiento normales y tienen consistencia de lectura tras la escritura. Si se produce un evento de conmutación por error entre la escritura y la lectura posterior, el sistema puede devolver brevemente una lectura que no tiene una consistencia alta. Todas las lecturas que se realizan desde una réplica de lectura tienen consistencia final y devuelven los datos en el mismo orden, a menudo con un retardo de réplica inferior a los 100 ms.

Preferencias de lectura de Amazon DocumentDB

Amazon DocumentDB permite configurar una opción de preferencia de lectura únicamente cuando los datos se leen del punto de conexión del clúster en modo de conjunto de réplicas. La configuración de una opción de preferencia de lectura afecta al modo en que el cliente o el controlador de MongoDB lee las solicitudes enviadas a las instancias del clúster de Amazon

DocumentDB. Puede definir opciones de preferencia de lectura para una consulta específica o como una opción general en el controlador de MongoDB. (Consulte la documentación del cliente o controlador para obtener instrucciones sobre cómo definir una opción de preferencia de lectura).

Si el cliente o controlador no se conecta a un punto de conexión del clúster de Amazon DocumentDB en modo de conjunto de réplicas, el resultado de especificar una preferencia de lectura es incierto.

Amazon DocumentDB no admite la configuración de conjuntos de etiquetas como preferencia de lectura.

Opciones de preferencias de lectura compatibles

- **`primary`**. especificar una preferencia de `primary` lectura ayuda a garantizar que todas las lecturas se dirijan a la instancia principal del clúster. Si la instancia principal no está disponible, se produce un error en la operación de lectura. Una preferencia de lectura `primary` proporciona coherencia de lectura tras escritura y es adecuada para aquellos casos de uso en los que la coherencia de lectura tras escritura tiene prioridad sobre la alta disponibilidad y el escalado de lectura.

  En el siguiente ejemplo se especifica una preferencia de lectura `primary`:

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

- **`primaryPreferred`**: si se especifica una preferencia de `primaryPreferred` lectura, las lecturas se redirigen a la instancia principal en condiciones normales de funcionamiento. Si se produce una conmutación por error de la instancia principal, el cliente envía las solicitudes a una réplica. Una preferencia de lectura `primaryPreferred` proporciona coherencia de lectura tras escritura en condiciones normales y una lectura consistente final durante un evento de conmutación por error. Una preferencia de lectura `primaryPreferred` es adecuada en aquellos casos de uso en los que la coherencia de lectura tras escritura tiene prioridad sobre el escalado de lectura, pero aún se requiere una alta disponibilidad.

  En el siguiente ejemplo se especifica una preferencia de lectura `primaryPreferred`:

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

- **`secondary`**: la especificación de una preferencia de `secondary` lectura garantiza que las lecturas solo se enruten a una réplica, nunca a la instancia principal. Si no hay instancias de réplica en el clúster, la solicitud de lectura produce un error. Una preferencia de lectura `secondary` proporciona lecturas consistentes finales y es adecuada en aquellos casos de uso en los que el rendimiento de escritura de la instancia principal tiene prioridad sobre la alta disponibilidad y la coherencia de lectura tras escritura.

  En el siguiente ejemplo se especifica una preferencia de lectura `secondary`:

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

- **`secondaryPreferred`**: la especificación de una preferencia de `secondaryPreferred` lectura garantiza que las lecturas se enruten a una réplica de lectura cuando hay una o más réplicas activas. Si no hay instancias de réplica activas en el clúster, la solicitud de lectura se envía a la instancia principal. Una preferencia de lectura `secondaryPreferred` proporciona operaciones de lectura consistente final cuando una réplica de lectura atiende la operación de lectura. Presenta lectura consistente tras escritura cuando es la instancia principal la que atiende la solicitud de lectura (salvo en los casos de conmutación por error). Una preferencia de lectura `secondaryPreferred` es adecuada para aquellos casos de uso en los que el escalado de lectura y la alta disponibilidad tienen prioridad sobre la coherencia de lectura tras escritura.

  En el siguiente ejemplo se especifica una preferencia de lectura `secondaryPreferred`:

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

- **`nearest`**: al especificar una preferencia de `nearest` lectura, las lecturas se distribuyen únicamente en función de la latencia medida entre el cliente y todas las instancias del clúster de Amazon DocumentDB. Una preferencia de lectura `nearest` proporciona operaciones de lectura consistente final cuando una réplica de lectura atiende la operación de lectura. Presenta lectura consistente tras escritura cuando es la instancia principal la que atiende la solicitud de lectura (salvo en los casos de conmutación por error). Una preferencia de lectura `nearest` es adecuada en aquellos casos de uso en los que conseguir la menor latencia de lectura posible y una alta disponibilidad tienen prioridad sobre la coherencia de lectura tras escritura y el escalado de lectura.

  En el siguiente ejemplo se especifica una preferencia de lectura `nearest`:

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

Alta disponibilidad

Amazon DocumentDB admite configuraciones de clúster altamente disponibles mediante el uso de réplicas como destinos de la conmutación por error de la instancia principal. Si la instancia principal produce un error, una réplica de Amazon DocumentDB pasará a ser la nueva instancia principal, con una breve interrupción durante la cual las solicitudes de lectura y escritura realizadas a la instancia principal producen una excepción.

Si el clúster de Amazon DocumentDB no contiene réplicas, se vuelve a crear la instancia principal cuando se produce un error. Dado que promover una réplica de Amazon DocumentDB es mucho más rápido que volver a crear la instancia principal, cree una o más réplicas de Amazon DocumentDB como destinos de conmutación por error.

Las réplicas que se han diseñado para utilizarlas como destinos de conmutación por error deben ser de la misma clase de instancia que la instancia principal. Deberían aprovisionarse en zonas de disponibilidad distintas de la principal. Puede controlar las réplicas que deben usarse como destinos de la conmutación por error. Para obtener instrucciones sobre cómo configurar Amazon DocumentDB para una alta disponibilidad, consulte Información general de la tolerancia a errores de clúster de Amazon DocumentDB.

Escalado de operaciones de lectura

Las réplicas de Amazon DocumentDB son ideales para el escalado de lectura. Están totalmente dedicadas a las operaciones de lectura en el volumen del clúster, es decir, las réplicas no procesan operaciones de escritura. La replicación de datos se produce en el volumen del clúster y no entre las instancias. Por lo tanto, los recursos de cada réplica se dedican a procesar las consultas, y no a replicar y escribir datos.

Si su aplicación necesita más capacidad de lectura, puede añadir una réplica al clúster rápidamente (normalmente en menos de 10 minutos). Si los requisitos de capacidad de lectura disminuyen, puede eliminar las réplicas que no necesite. Con las réplicas de Amazon DocumentDB, solo paga por la capacidad de lectura que necesite.

Amazon DocumentDB permite el escalado de lectura en el cliente mediante el uso de opciones de preferencia de lectura. Para obtener más información, consulte Preferencias de lectura de Amazon DocumentDB.

Eliminaciones de TTL

Las eliminaciones de un área de índice TTL logradas a través de un proceso en segundo plano son el mejor esfuerzo y no están garantizadas dentro de un período de tiempo específico. Factores como el tamaño de instancia, la utilización de recursos de instancia, el tamaño del documento y el rendimiento general pueden afectar la temporización de una eliminación de TTL.

Cuando el monitor TTL elimina sus documentos, cada eliminación incurre en costos de E/S, lo que aumentará su factura. Si el rendimiento y las tasas de eliminación de TTL aumentan, debería esperar un aumento en su factura debido al aumento del uso de E/S.

Al crear un índice TTL en una colección existente, debe eliminar todos los documentos caducados antes de crear el índice. La implementación actual del TTL está optimizada para eliminar una pequeña fracción de los documentos de la colección, lo que suele ocurrir si el TTL estaba activado en la colección desde el principio, y puede resultar en un aumento de las IOPS de lo necesario si es necesario eliminar un gran número de documentos de una sola vez.

En lugar de utilizar un índice TTL para eliminar documentos, puede segmentar documentos en colecciones según el tiempo y simplemente eliminar esas colecciones cuando los documentos ya no sean necesarios. Por ejemplo: puede crear una colección por semana y eliminarla sin incurrir en costos de E/S. Esto puede resultar considerablemente más rentable que utilizar un índice TTL.

Recursos facturables

Identificación de los recursos facturables de Amazon DocumentDB

Como servicio de bases de datos totalmente administrado, Amazon DocumentDB cobra por las instancias, el almacenamiento I/Os, las copias de seguridad y la transferencia de datos. Para obtener más información, consulte Amazon DocumentDB (with MongoDB compatibility) Pricing.

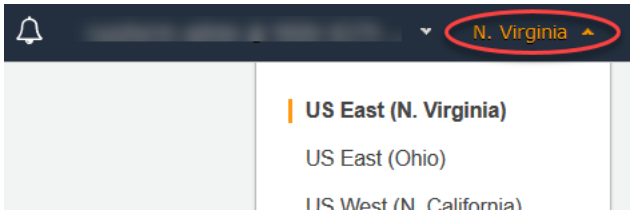
Para descubrir los recursos facturables de su cuenta y, si es posible, eliminarlos, puede usar el Consola de administración de AWS comando o. AWS CLI

Uso de Consola de administración de AWS

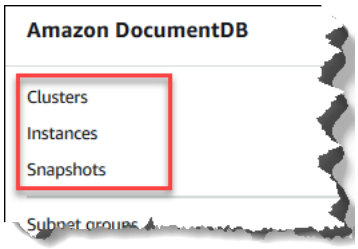
Con el Consola de administración de AWS, puede descubrir los clústeres, las instancias y las instantáneas de Amazon DocumentDB que ha aprovisionado para un determinado. Región de AWS

Para detectar clústeres, instancias e instantáneas

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. Para descubrir los recursos facturables en una región que no sea la región predeterminada, en la esquina superior derecha de la pantalla, seleccione la región en la Región de AWS que desee buscar.



3. En el panel de navegación, seleccione el tipo de recurso facturable que le interesa: Clusters (Clústeres), Instances (Instancias) o Snapshots (Instantáneas).



4. Todos los clústeres, instancias o instantáneas aprovisionados para la región se indican en el panel de la derecha. Se le cobrará por los clústeres, las instancias y las instantáneas.

Uso de AWS CLI

Con el AWS CLI, puede descubrir los clústeres, instancias e instantáneas de Amazon DocumentDB que ha aprovisionado para un determinado lugar. Región de AWS

Para detectar clústeres e instancias

El código siguiente muestra todos los clústeres e instancias para la región especificada. Si desea buscar clústeres e instancias en la región predeterminada, puede omitir el parámetro `--region`.

Example

Para Linux, macOS o Unix:

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

```
    --query 'DBClusters[?Engine==`docdb`]' | \
       grep -e "DBClusterIdentifier" -e "DBInstanceIdentifier"
```

Para Windows:

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

La salida de esta operación será similar a lo que se indica a continuación.

```
"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",
```

Para detectar instantáneas

El código siguiente muestra todas las instantáneas para la región especificada. Si desea buscar instantáneas en la región predeterminada, puede omitir el parámetro `--region`.

Para Linux, macOS o Unix:

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

Para Windows:

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

La salida de esta operación será similar a lo que se indica a continuación.

```
[
```

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

Solo tiene que eliminar las instantáneas `manual`. Las instantáneas `Automated` se eliminan cuando elimina el clúster.

Eliminación de recursos facturables no deseados

Para eliminar un clúster, antes debe eliminar todas las instancias de ese clúster.

- Para eliminar las instancias, consulte Eliminación de una instancia de Amazon DocumentDB.

> **Important**
>
> Aunque elimine las instancias de un clúster, se le seguirá facturando por el uso del almacenamiento y las copias de seguridad asociadas a ese clúster. Para evitar todos los cargos, también debe eliminar el clúster y las instantáneas manuales.

- Para eliminar clústeres, consulte Eliminación de un clúster de Amazon DocumentDB.
- Para eliminar instantáneas manuales, consulte Eliminación de una instantánea del clúster.

¿Qué es una base de datos de documentos?

Algunos desarrolladores no piensan en su modelo de datos en términos de filas y columnas normalizadas. Normalmente, en la capa de aplicación, los datos se representan como un documento JSON, porque es más intuitivo para los desarrolladores pensar en su modelo de datos como un documento.

La popularidad de las bases de datos documentales ha aumentado porque permiten conservar los datos en una base de datos utilizando el mismo formato de modelo de documentos que se usa en el código de la aplicación. Las bases de datos documentales proporcionan API eficaces e intuitivas que permiten un desarrollo flexible y ágil.

Temas

- Casos de uso de bases de datos documentales
- Descripción de documentos
- Trabajo con documentos

Casos de uso de bases de datos documentales

La necesidad de usar una base de datos de documentos u otro tipo de base de datos para administrar los datos dependerá del caso de uso. Las bases de datos documentales son útiles para cargas de trabajo que requieren un esquema flexible que permita un desarrollo rápido e iterativo. A continuación, se incluyen algunos ejemplos de casos de uso para los que las bases de datos documentales pueden ofrecer importantes ventajas:

Temas

- Perfiles de usuario
- Real-time macrodatos
- Administración de contenido

Perfiles de usuario

Como las bases de datos documentales tienen un esquema flexible, pueden almacenar documentos que tengan atributos y valores de datos diferentes. Las bases de datos documentales son una solución práctica para los perfiles online en los que diferentes usuarios proporcionan diferentes tipos de información. Mediante una base de datos de documentos, puede almacenar cada perfil de usuario de forma eficaz almacenando solo los atributos que son específicos de cada usuario.

Suponga que un usuario decide añadir o eliminar la información de su perfil. En este caso, su documento podría reemplazarse fácilmente por una versión actualizada que contuviera los atributos y datos recién añadidos u omitir todos los atributos y datos recién omitidos. Las bases de datos documentales administran fácilmente este nivel de detalle y fluidez.

Real-time macrodatos

Históricamente, la capacidad de extraer información de los datos operativos se veía obstaculizada por el hecho de que las bases de datos operativas y las bases de datos analíticas se mantenían en entornos diferentes: operativos y, respectivamente. business/reporting Ser capaces de extraer

información operativa en tiempo real es fundamental en un entorno empresarial altamente competitivo. Mediante el uso de bases de datos documentales, una empresa puede almacenar y administrar datos operativos de cualquier origen e incluir los datos de forma simultánea en el motor de BI elegido para su análisis. No es necesario tener dos entornos.

Administración de contenido

Para administrar eficazmente el contenido, debe poder recopilar y agrupar contenido de una variedad de orígenes y enviárselo al cliente. Debido a su esquema flexible, las bases de datos documentales son perfectas para recopilar y almacenar cualquier tipo de datos. Puede utilizarlas para crear e incorporar nuevos tipos de contenido, incluido el contenido generado por el usuario, como imágenes, comentarios, y vídeos.

Descripción de documentos

Las bases de datos de documentos se utilizan para almacenar datos semiestructurados como un documento en lugar de normalizar los datos entre varias tablas, cada una con una estructura única y fija, como en una base de datos relacional. Los documentos almacenados en una base de datos de documentos usan pares de clave-valor anidados para proporcionar la estructura o el esquema del documento. Sin embargo, se pueden almacenar diferentes tipos de documentos en la misma base de datos de documentos, cumpliendo así el requisito de procesar datos similares que tienen diferentes formatos. Por ejemplo, dado que cada documento se describe a sí mismo, los JSON-encoded documentos de una tienda en línea que se describen en el tema se Documentos de ejemplo en una base de datos de documentos pueden almacenar en la misma base de datos de documentos.

Temas

- Terminología SQL frente a terminología no relacional
- Documentos simples
- Documentos incrustados
- Documentos de ejemplo en una base de datos de documentos
- Descripción de la normalización en una base de datos de documentos

Terminología SQL frente a terminología no relacional

En la tabla siguiente se compara la terminología utilizada en las bases de datos documentales (MongoDB) con la terminología utilizada en las bases de datos SQL.

SQL	MongoDB
Tabla	Recopilación
Fila	Documento
Columna	Campo
Clave principal	ObjectId
Índice	Índice
Visualización	Visualización
Tabla u objeto anidado	Documento incrustado
Matriz	Matriz

Documentos simples

Todos los documentos de una base de datos de documentos son autodescriptivos. En esta documentación se utilizan documentos JSON-like formateados, aunque puede utilizar otros medios de codificación.

Un documento simple tiene uno o varios campos que están en el mismo nivel del documento. En el siguiente ejemplo, los campos `SSN`, `LName`, `FName`, `DOB`, `Street`, `City`, `State-Province`, `PostalCode` y `Country` son todos elementos del mismo nivel en el 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"
}
```

Cuando la información está organizada en un documento simple, cada campo se administra por separado. Para recuperar la dirección de una persona, debe recuperar `Street`, `City`, `State-Province`, `PostalCode` y `Country` como elementos de datos individuales.

Documentos incrustados

Un documento complejo organiza sus datos mediante la creación de documentos incrustados en el documento. Los documentos incrustados ayudan a administrar los datos en grupos y como elementos de datos individuales, lo que sea más eficaz para cada caso. Utilizando el ejemplo anterior, podría incrustar un documento `Address` en el documento principal, lo que produciría la siguiente estructura de documentos:

```
{
   "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"
   }
}
```

Ahora puede obtener acceso a los datos del documento como campos individuales (`"SSN":`), como un documento incrustado (`"Address":`) o como un miembro de un documento incrustado (`"Address":{"Street":}`).

Documentos de ejemplo en una base de datos de documentos

Como se indicó anteriormente, puesto que cada documento de una base de datos de documentos es autodescriptivo, la estructura de los documentos de una base de datos de documentos puede ser diferente. Los siguientes dos documentos, uno para un libro y otro para una publicación periódica, son diferentes estructuralmente. Sin embargo, ambos pueden estar en la misma base de datos de documentos.

A continuación, se muestra un documento de libro de ejemplo:

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

A continuación, se muestra un documento de publicación periódica de ejemplo con dos artículos:

```
{
    "_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"
```

```
}
```

Compare la estructura de estos dos documentos. Con una base de datos relacional, necesita separar las tablas "periodical" y "books", o una sola tabla con los campos no utilizados, como "Publication," "Issue," "Articles" y "MI", como valores `null`. Como las bases de datos de documentos son semiestructuradas, en las que cada documento define su propia estructura, estos dos documentos pueden coexistir en la misma base de datos de documentos sin campos `null`. Las bases de datos documentales son adecuadas para trabajar con datos dispersos.

El desarrollo de una base de datos de documentos permite el desarrollo rápido e iterativo. Esto se debe a que puede cambiar la estructura de los datos de un documento de forma dinámica, sin tener que cambiar el esquema de toda la colección. Las bases de datos documentales están especialmente indicadas para un desarrollo ágil y para entornos que cambian dinámicamente.

Descripción de la normalización en una base de datos de documentos

Las bases de datos documentales no están normalizadas; los datos que se encuentran en un documento se pueden repetir en otro documento. Asimismo, puede haber discrepancias de datos entre los documentos. Considere, por ejemplo, un escenario en el que realiza una compra en una tienda online y todos los detalles de sus compras se almacenan en un único documento. El documento podría tener un aspecto similar al siguiente documento JSON:

```
{
    "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"
}
```

Toda esta información se almacena como un documento en una colección de transacciones. Posteriormente, se da cuenta de que ha olvidado comprar un artículo. Así que vuelve a iniciar sesión en la misma tienda y realiza otra compra, que también se almacena como otro documento en la colección de transacciones.

```
{
    "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"
}
```

Observe la redundancia entre estos dos documentos: su nombre y el ID de comprador (y, si utilizó la misma tarjeta de crédito, la información de la tarjeta de crédito). Pero eso no es un problema, porque el almacenamiento es barato y cada documento registra completamente una transacción que se puede recuperar rápidamente con una sencilla consulta de clave-valor que no requiere uniones.

También hay una discrepancia aparente entre los dos documentos: la información de su tarjeta de crédito. Esta solo es una discrepancia aparente, ya que es probable que utilizara una tarjeta de crédito diferente para cada compra. Cada documento es correcto para la transacción que documenta.

Trabajo con documentos

Como base de datos de documentos, Amazon DocumentDB facilita el almacenamiento, la consulta y la indexación de datos JSON. En Amazon DocumentDB, una colección es similar a una tabla de una base de datos relacional, con la salvedad de que no se aplica ningún esquema a todos los documentos. Las colecciones le permiten agrupar documentos similares manteniéndolos en la misma base de datos, sin necesidad de que tengan la misma estructura.

Sigamos con los documentos de ejemplo de las secciones anteriores y suponga ahora que tiene colecciones para `reading_material` y `office_supplies`. Es responsabilidad de su software decidir a qué colección pertenece un documento.

En los siguientes ejemplos se utiliza la API de MongoDB para mostrar cómo añadir, consultar, actualizar y eliminar documentos.

Temas

- Adición de documentos
- Consulta de documentos
- Actualización de documentos
- Eliminación de documentos

Adición de documentos

En Amazon DocumentDB, se crea una base de datos cuando se agrega por primera vez un documento a una colección. En este ejemplo, está creando una colección denominada `example` en la base de datos `test`, que es la base de datos predeterminada cuando se conecta a un clúster. Dado que la conexión se crea implícitamente cuando se inserta el primer documento, no hay comprobación de errores del nombre de la colección. Por lo tanto, un error tipográfico en el nombre de la colección, como `eexample` en lugar de `example`, creará y agregará el documento a la colección `eexample` en lugar de la colección prevista. La comprobación de errores debe realizarla su aplicación.

En los ejemplos siguientes, se utiliza la API de MongoDB para añadir documentos.

Temas

- Adición de un solo documento
- Adición de varios documentos

Adición de un solo documento

Para añadir un único documento a una colección, utilice la operación `insertOne( {} )` con el documento que desea añadir a la colección.

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

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

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

Adición de varios documentos

Para añadir varios documentos a una colección, utilice la operación `insertMany( [{},..., {}] )` con una lista de los documentos que desea añadir a la colección. Aunque los documentos de esta lista en particular tengan diferentes esquemas, se pueden añadir a la misma colección.

```
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
            }
        }
    ]
)
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

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

Consulta de documentos

A veces, es posible que tenga que examinar el inventario de su tienda online para que los clientes puedan ver y comprar lo que usted vende. Consultar una colección es relativamente fácil, tanto si desea consultar todos los documentos de la colección como solo aquellos que cumplan un determinado criterio.

Para consultar documentos, utilice la operación `find()`. El comando `find()` tiene un único parámetro de documento que define los criterios que se utilizan al elegir los documentos que se devuelven. El resultado de `find()` es un documento formateado como una sola línea de texto sin saltos de línea. Para formatear el documento resultante para facilitar su lectura, utilice `find().pretty()`. En todos los ejemplos que se muestran en este tema, se utiliza `.pretty()` para formatear la salida.

Utilice los cuatro documentos que insertó en la colección `example` de los dos ejercicios anteriores `insertOne()` e `insertMany()`.

Temas

- Recuperar todos los documentos de una colección
- Recuperación de documentos que coincidan con un valor de campo
- Recuperación de documentos que coincidan con un documento incrustado
- Recuperación de documentos que coincidan con un valor de campo de un documento incrustado
- Recuperación de documentos que coincidan con una matriz
- Recuperación de documentos que coincidan con un valor de una matriz
- Recuperación de documentos mediante operadores

Recuperar todos los documentos de una colección

Para recuperar todos los documentos de la colección, utilice la operación `find()` con un documento de consulta vacío.

La siguiente consulta devuelve todos los documentos de la colección `example`.

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

Recuperación de documentos que coincidan con un valor de campo

Para recuperar todos los documentos que coincidan con un campo y valor, utilice la operación `find()` con un documento de consulta que identifique los campos y valores que desee.

Si se utilizan los documentos anteriores, esta consulta devuelve todos los documentos cuyo campo "Item" contiene "Pen".

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

Recuperación de documentos que coincidan con un documento incrustado

Para buscar todos los documentos que coinciden con un documento incrustado, utilice la operación `find()` con un documento de consulta que especifique el nombre del documento incrustado y todos los campos y los valores de ese documento incrustado.

Cuando se buscan coincidencias con un documento incrustado, el documento incrustado del documento debe tener el mismo nombre que en la consulta. Además, los campos y los valores del documento incrustado deben coincidir con la consulta.

La siguiente consulta devuelve únicamente el documento "Poster Paint". Esto se debe a que "Pen" tiene diferentes valores para "`MinOnHand`" y "`OnHand`", y "Spray Paint" tiene un campo más (`OrderQnty`) que el documento de consulta.

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

Recuperación de documentos que coincidan con un valor de campo de un documento incrustado

Para buscar todos los documentos que coinciden con un documento incrustado, utilice la operación `find()` con un documento de consulta que especifique el nombre del documento incrustado y todos los campos y los valores de ese documento incrustado.

Dados los documentos anteriores, la siguiente consulta utiliza la "notación de puntos" para especificar el documento incrustado y los campos de interés. Se devolverá cualquier documento que coincida con ellos, independientemente de los otros campos que puedan existir en el documento incrustado. La consulta devuelve "Poster Paint" y "Spray Paint", ya que ambos coinciden con los campos y los valores especificados.

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

Recuperación de documentos que coincidan con una matriz

Para buscar todos los documentos que coincidan con una matriz, utilice la operación `find()` con el nombre de la matriz que le interese y todos los valores de esa matriz. La consulta devuelve todos los documentos que tengan una matriz con ese nombre y cuyos valores de la matriz sean idénticos y estén en el mismo orden que en la consulta.

La siguiente consulta devuelve únicamente "Pen", ya que "Poster Paint" tiene un color adicional (White) y "Spray Paint" tiene los colores en otro orden.

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

Recuperación de documentos que coincidan con un valor de una matriz

Para buscar todos los documentos que tengan un valor determinado en una matriz, utilice la operación `find()` con el nombre de la matriz y el valor que le interese.

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

La operación anterior devuelve los tres documentos, ya que cada uno de ellos tiene una matriz denominada `Colors` y el valor "`Red`" en algún lugar de la matriz. Si especifica el valor "`White`", la consulta solo devolvería "Poster Paint".

Recuperación de documentos mediante operadores

La siguiente consulta devuelve todos los documentos en los que el valor de "`Inventory.OnHand`" es menor que 50.

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

Para obtener una lista de los operadores de consulta admitidos, consulte Operadores de consulta y proyección.

Actualización de documentos

Normalmente, los documentos no son estáticos y se actualizan como parte de los flujos de trabajo de la aplicación. En los siguientes ejemplos se muestran algunas de las formas en que puede actualizar los documentos.

Para actualizar un documento existente, use la operación `update()`. La operación `update()` tiene dos parámetros de documento. El primer documento identifica el documento o los documentos que se van a actualizar. El segundo documento especifica las actualizaciones que se deben realizar.

Al actualizar un campo existente ya sea un campo sencillo, una matriz o un documento incrustado debe especificar el nombre del campo y sus valores. Al final de la operación, es como si el campo del documento antiguo se ha sustituido por el campo y los valores nuevos.

Temas

- Actualizar los valores de un campo existente

- Agregado de un campo nuevo
- Sustitución de un documento incrustado
- Inserción de campos nuevos en un documento incrustado
- Eliminación de un campo de un documento
- Eliminación de un campo de varios documentos

Actualizar los valores de un campo existente

Utilice los cuatro documentos siguientes que ha añadido anteriormente para las siguientes operaciones de actualización.

```
{
    "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
    }
}
```

Para actualizar un campo sencillo

Para actualizar un campo sencillo, utilice `update()` con `$set` para especificar el nombre del campo y el valor nuevo. En el siguiente ejemplo, se cambia `Item` de "Pen" a "Gel Pen".

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

Los resultados de esta operación serán similares a lo que se indica a continuación (formato JSON).

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

Para actualizar una matriz

En el siguiente ejemplo, se sustituye la gama de colores existente por una matriz nueva que incluye `Orange` y elimina `White` de la lista de colores. La lista de colores nueva se encuentra en el orden especificado en la operación `update()`.

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

Los resultados de esta operación serán similares a lo que se indica a continuación (formato JSON).

```
{
    "Item": "Poster Paint",
```

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

Agregado de un campo nuevo

Para modificar un documento añadiendo uno o varios campos nuevos, utilice la operación `update()` con un documento de consulta que identifique el documento que se va a insertar y los nuevos campos y valores que se van a insertar mediante el operador `$set`.

En el siguiente ejemplo, se añade el campo `UnitPrice` con el valor `3.99` en el documento Spray Paints. Tenga en cuenta que el valor `3.99` es numérico y no una cadena.

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

Los resultados de esta operación serán similares a lo que se indica a continuación (JSON format).

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

Sustitución de un documento incrustado

Para modificar un documento sustituyendo un documento incrustado, utilice la operación `update()` con documentos que identifiquen el documento incrustado y sus campos y valores nuevos mediante el operador `$set`.

Dado el documento siguiente.

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

Para sustituir un documento incrustado

En el ejemplo siguiente, se sustituye el documento Date actual por uno nuevo que solo tiene los campos `Month` y `Day` en el que se ha eliminado `Year`.

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

Los resultados de esta operación serán similares a lo que se indica a continuación (formato JSON).

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

Inserción de campos nuevos en un documento incrustado

Para añadir campos a un documento incrustado

Para modificar un documento añadiendo uno o varios campos nuevos a un documento incrustado, utilice la operación `update()` con documentos que identifiquen el documento incrustado y la "notación de puntos" para especificar el documento incrustado y los campos y valores nuevos que se van a insertar mediante el operador `$set`.

Teniendo en cuenta el siguiente documento, el código siguiente utiliza la "notación de puntos" para insertar los campos `Year` y `DoW` en el documento incrustado `Date` y `Words` en el documento principal.

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

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

Los resultados de esta operación serán similares a lo que se indica a continuación (formato JSON).

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

Eliminación de un campo de un documento

Para modificar un documento eliminando uno se sus campos, utilice la operación `update()` con un documento de consulta que identifique el documento cuyo campo se va a eliminar y el operador `$unset` para especificar campo que se va a eliminar.

En el siguiente ejemplo, se elimina el campo `Words` del documento anterior.

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

Los resultados de esta operación serán similares a lo que se indica a continuación (formato JSON).

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

Eliminación de un campo de varios documentos

Para modificar un documento mediante la eliminación de un campo de varios documentos, utilice la operación `update()` con el operador `$unset` y la opción `multi` establecida en `true`.

En el ejemplo siguiente se quita el campo `Inventory` de todos los documentos de la colección de ejemplo. Si un documento no tiene el campo `Inventory`, no se realiza ninguna acción en ese documento. Si `multi: true` se omite, la acción solo se realiza en el primer documento que cumple el criterio.

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

Eliminación de documentos

Para eliminar un documento de la base de datos, utilice la operación `remove()`, especificando el documento que se va a eliminar. El código siguiente elimina "Gel Pen" de la colección `example`.

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

Para eliminar todos los documentos de la base de datos, use la operación `remove()` con una consulta vacía.

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

Introducción a Amazon DocumentDB

Existen muchas formas de conectarse a Amazon DocumentDB y empezar a usarlo. Creamos esta guía porque nos pareció que era la forma más rápida, sencilla y fácil de que los usuarios comenzaran a utilizar nuestra potente base de datos de documentos. En esta guía se utiliza el AWS CloudShell para conectarse al clúster de Amazon DocumentDB directamente desde la Consola de administración de AWS y consultarlo. Los nuevos clientes que cumplan los requisitos para la capa AWS gratuita pueden usar Amazon DocumentDB de forma CloudShell gratuita. Si su AWS CloudShell entorno o clúster de Amazon DocumentDB utiliza recursos que van más allá de la capa gratuita, se le cobrarán las AWS tarifas normales por esos recursos. Esta guía le permitirá empezar a utilizar Amazon DocumentDB en menos de cinco minutos.

> **Note**
>
> Las instrucciones de esta guía son específicas para crear clústeres basados en instancias de Amazon DocumentDB y conectarse a ellos cuando Amazon DocumentDB y yo estén disponibles. AWS CloudShell
>
> - Si desea crear clústeres elásticos de Amazon DocumentDB y conectarse a ellos, consulte Introducción a los clústeres elásticos de Amazon DocumentDB.
> - Si se encuentra en las regiones de AWS China, consulte. Conexión automática de Amazon EC2

Temas

- Requisitos previos
- Paso 1: creación de un clúster
- Paso 2: Conéctese a su clúster
- Paso 3: Insertar y consultar datos
- Paso 4: Explorar
- Inicio rápido de Amazon DocumentDB a usar CloudFormation

Requisitos previos

Antes de crear el primer clúster de Amazon DocumentDB, debe hacer lo siguiente:

Creación de una cuenta de Amazon Web Services (AWS)

Para empezar a utilizar Amazon DocumentDB, debe tener una cuenta de Amazon Web Services (AWS). La AWS cuenta es gratuita. Solo se paga por los servicios y los recursos que se utilicen.

Si no tiene una Cuenta de AWS, complete los siguientes pasos para crear una.

Para inscribirse en un Cuenta de AWS

1. https://signin.aws.amazon.com/signup¿Abierto? request_type=registrarse.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica o mensaje de texto e indicar un código de verificación en el teclado del teléfono.

   Cuando te registras en un, se crea un Cuenta de AWS. Usuario raíz de la cuenta de AWS El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar Tareas que requieren acceso de usuario raíz.

Configura los permisos necesarios AWS Identity and Access Management (IAM).

El acceso para administrar los recursos de Amazon DocumentDB, como los clústeres, las instancias y los grupos de parámetros de clústeres, requiere credenciales que AWS se puedan utilizar para autenticar las solicitudes. Para obtener más información, consulte Identity and Access Management para Amazon DocumentDB.

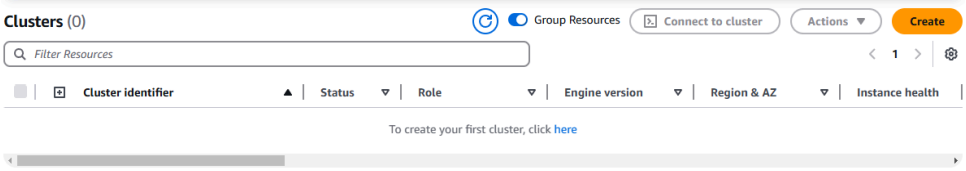
1. En la barra de búsqueda de Consola de administración de AWS, introduzca **`IAM`** y seleccione IAM en el menú desplegable que aparece.
2. Cuando esté en la consola de IAM, seleccione Usuarios en el panel de navegación.
3. Seleccione su nombre de usuario.
4. Elija Añadir permisos.
5. Seleccione Adjuntar políticas directamente.
6. Escriba `AmazonDocDBFullAccess` en la barra de búsqueda y selecciónelo en cuanto aparezca en los resultados de búsqueda.
7. Elija Siguiente.
8. Elija Añadir permisos.

> **Note**
>
> Tu AWS cuenta incluye una VPC predeterminada en cada región. Si decide utilizar una Amazon VPC, complete los pasos del tema Creación de una Amazon VPC en la Guía del usuario de Amazon VPC.

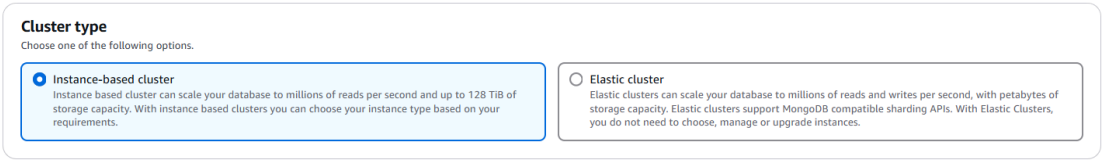
Paso 1: creación de un clúster

En este paso creará un clúster de Amazon DocumentDB.

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En la consola de administración de Amazon DocumentDB, en Clústeres, elija Crear.



3. En la página Crear un clúster de Amazon DocumentDB, en la sección Tipo de clúster, elija Instance-based clúster (es la opción predeterminada).

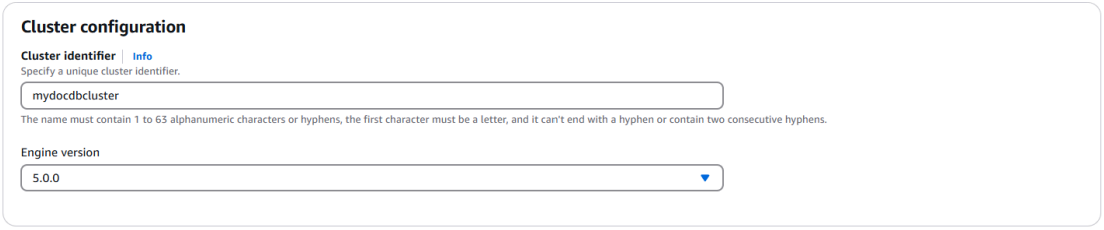


> **Note**
>
> La otra opción de esta categoría es el clúster elástico. Para obtener más información sobre los clústeres elásticos de Amazon DocumentDB, consulte Uso de clústeres elásticos de Amazon DocumentDB.

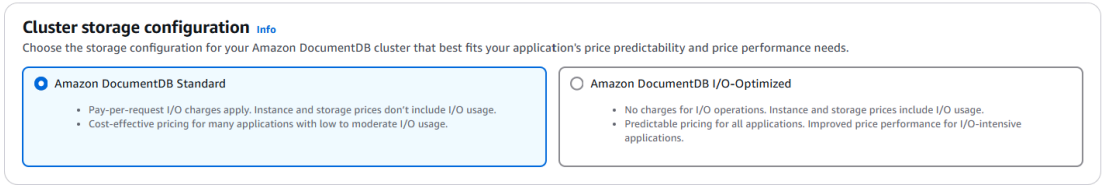
4. En la sección Configuración de clúster:

   a. En el campo Identificador del clúster, ingrese un nombre único, como **`mydocdbcluster`**. Tenga en cuenta que la consola cambiará todos los nombres de los clústeres a minúsculas independientemente de cómo se ingresen.

   b. En Versión del motor, elija 5.0.0.



5. En la sección Configuración del almacenamiento en clúster, elija Amazon DocumentDB estándar (es la opción predeterminada).



> **Note**
>
> La otra opción de esta categoría es Amazon I/O-Optimized DocumentDB. Para obtener más información sobre las opciones, consulte Configuraciones de almacenamiento en clústeres de Amazon DocumentDB.

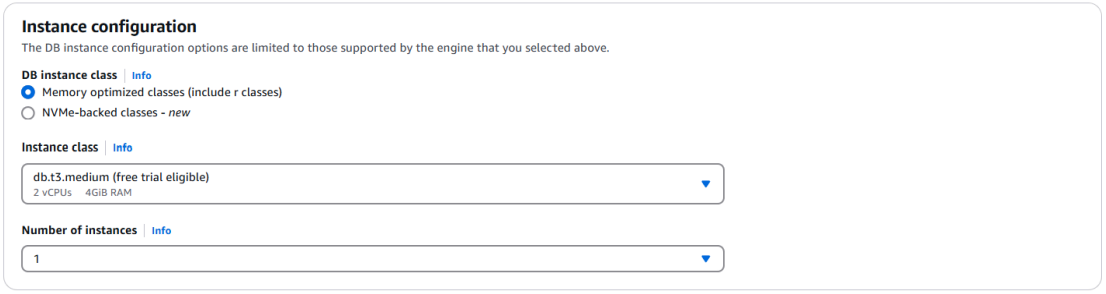
6. Sección de Configuración de instancias:

   a. En Clase de instancia de base de datos, elija Clases optimizadas para memoria (incluidas las clases r), que es la opción predeterminada.

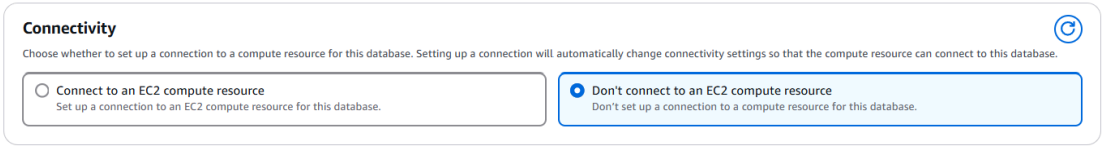
      La otra opción de instancia son NVMe-backed las clases. Para obtener más información, consulte NVMe-backed instancias.

   b. En Clase de instancia, elija db.t3.medium. Esta opción es apta para la prueba AWS gratuita.

   c. En Número de instancias, elija 1 instancia. La elección de una instancia ayuda a minimizar los costos. Si se tratara de un sistema de producción, le recomendamos que aprovisione tres instancias para una alta disponibilidad.

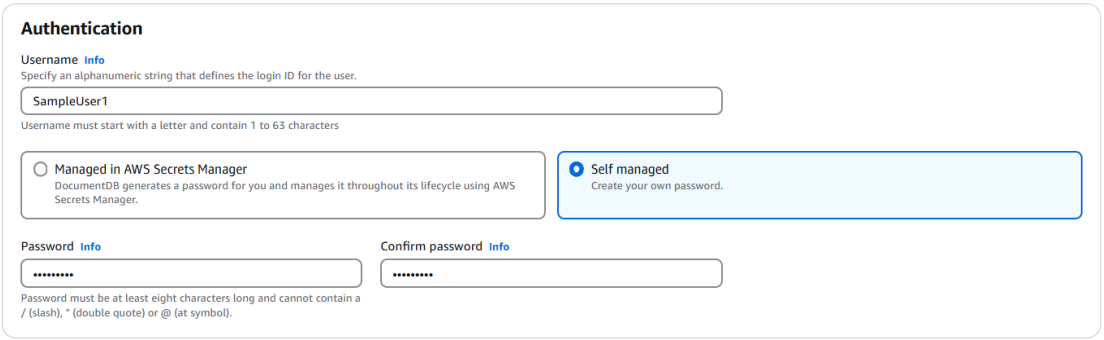


7. En la sección Conectividad, deje la configuración predeterminada de No conectarse a un recurso de computación de EC2.



8. En la sección Autenticación, ingrese un nombre de usuario para el usuario principal y, a continuación, elija Autoadministrado. Ingrese una contraseña y, a continuación, confírmela.

   Si, por el contrario, ha elegido Gestionado en AWS Secrets Manager, consulte Administración de contraseñas con Amazon DocumentDB y AWS Secrets Manager para obtener más información.



9. Deje todas las demás opciones en sus valores predeterminados y seleccione Crear clúster.

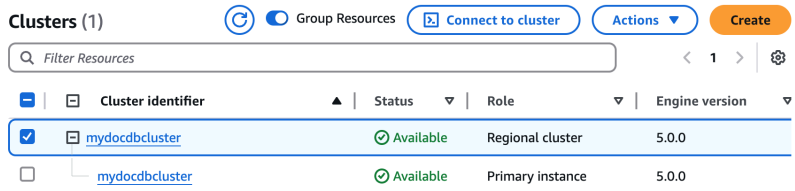
Amazon DocumentDB está aprovisionando su clúster, lo que puede tardar unos minutos en terminar.

> **Note**
>
> Para obtener información sobre los valores de estado de los clústeres, consulte Valores de estado del clúster en el capítulo Cómo monitorizar Amazon DocumentDB.

Paso 2: Conéctese a su clúster

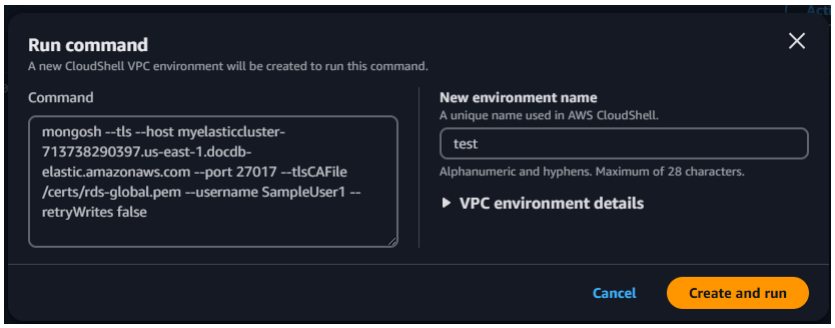
Conéctese a su clúster de Amazon DocumentDB mediante AWS CloudShell.

1. En la consola de administración de Amazon DocumentDB, en Clústeres, localice el clúster que ha creado. Para elegir el clúster, haga clic en la casilla de verificación situada junto a él.



2. Elija Conectarse al clúster (que se encuentra junto al menú desplegable Acciones). Este botón se activa solamente después de haber hecho clic en la casilla de verificación situada junto al clúster, y el estado del clúster regional y las instancias principales se muestra como Disponible. Aparece la pantalla de comandos CloudShell Ejecutar.

3. En el campo Nombre del nuevo entorno, ingrese un nombre único, como “prueba” y elija Crear y ejecutar. Los detalles del entorno de la VPC se configuran automáticamente para la base de datos de Amazon DocumentDB.



4. Cuando vea la petición, introduzca la contraseña que creó en el Paso 1: crear un clúster de Amazon DocumentDB (subpaso 7).



Después de ingresar la contraseña, cuando la petición se convierte en `rs0 [direct: primary] <env-name>>`, significa que se ha conectado correctamente al clúster de Amazon DocumentDB.

> **Note**
>
> Para obtener información sobre la solución de problemas, consulte Solución de problemas de Amazon DocumentDB.

Paso 3: Insertar y consultar datos

Ahora que está conectado a su clúster, puede realizar algunas consultas para familiarizarse con el uso de una base de datos de documentos.

1. Para insertar un solo documento, escriba lo siguiente:

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

   Obtiene el siguiente resultado:

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

2. Puede leer el documento que escribió con el comando `findOne()` (ya que solo devuelve un documento). La siguiente entrada:

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

   Obtiene el siguiente resultado:

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

3. Para realizar algunas consultas más, plantéese un caso de uso de perfiles de juegos. Primero, inserte algunas entradas en una colección titulada `profiles`. La siguiente entrada:

```
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 }
])
```

Obtiene el siguiente resultado:

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

4. Utilice el comando `find()` para devolver todos los documentos de la colección de perfiles. La siguiente entrada:

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

Obtendrá un resultado que coincidirá con los datos que escribió en el paso 3.

5. Utilice una consulta para un único documento mediante un filtro. La siguiente entrada:

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

Obtiene el siguiente resultado:

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

6. Ahora intentemos buscar un perfil y modificarlo con el comando `findAndModify`. Le daremos al usuario Matt 10 puntos adicionales con el siguiente código:

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

Se obtiene el siguiente resultado (tenga en cuenta que la puntuación aún no ha aumentado):

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

7. Puede comprobar que su puntuación ha cambiado con la siguiente consulta:

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

   Obtiene el siguiente resultado:

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

Paso 4: Explorar

¡Enhorabuena! Ha completado correctamente la Guía de introducción a clústeres basados en instancias de Amazon DocumentDB.

Pasos siguientes Descubra cómo aprovechar al máximo esta base de datos con algunas de sus características más populares:

- Administración de Amazon DocumentDB
- Escalado
- Copia de seguridad y restauración

> **Note**
>
> El clúster que creó a partir de este ejercicio de introducción seguirá acumulando costos a menos que lo elimine. Para obtener instrucciones, consulte Cómo eliminar un clúster de Amazon DocumentDB.

Inicio rápido de Amazon DocumentDB a usar CloudFormation

Esta sección contiene pasos y otra información que le ayudarán a comenzar rápidamente a utilizar Amazon DocumentDB (compatible con MongoDB) mediante CloudFormation. Para obtener información general acerca de Amazon DocumentDB, consulte Qué es Amazon DocumentDB (compatible con MongoDB).

Estas instrucciones utilizan una AWS CloudFormation plantilla para crear un clúster e instancias en su Amazon VPC predeterminada. Para obtener instrucciones sobre cómo crear estos recursos usted mismo, consulte Introducción a Amazon DocumentDB.

> **Important**
>
> La CloudFormation pila que crea esta plantilla crea varios recursos, incluidos los recursos de Amazon DocumentDB (por ejemplo, un clúster y unas instancias) y Amazon Elastic Compute Cloud (por ejemplo, un grupo de subredes).
> Algunos de estos recursos no se incluyen en la capa gratuita. Para obtener más información, consulte Precios de Amazon DocumentDB y Precios de Amazon EC2. Puede eliminar la pila cuando haya terminado de utilizarla para suspender todos los cargos.

Esta CloudFormation pila está pensada únicamente para fines didácticos. Si utilizas esta plantilla para un entorno de producción, utiliza políticas de IAM y seguridad más estrictas. Para obtener información sobre cómo proteger los recursos, consulte Seguridad de Amazon VPC y Seguridad y redes de Amazon EC2.

Temas

- Requisitos previos
- Lanzamiento de una Amazon DocumentDB CloudFormation pila
- Acceso al clúster de Amazon DocumentDB
- Protección de terminación y protección de eliminación

Requisitos previos

Antes de crear un clúster de Amazon DocumentDB, debe hacer lo siguiente:

- Una VPC predeterminada de Amazon
- Los permisos necesarios de IAM

Permisos de IAM necesarios

Los siguientes permisos permiten crear recursos para la pila de CloudFormation :

AWS Políticas administradas

- `AWSCloudFormationReadOnlyAccess`
- `AmazonDocDBFullAccess`

Permisos de IAM adicionales

La siguiente política describe los permisos adicionales que se requieren para crear y eliminar esta CloudFormation pila.

En los siguientes ejemplos, sustituya cada uno *`user input placeholder`* por la información de su recurso.

Pares de claves de Amazon EC2

Debe tener un par de claves (y el archivo PEM) disponibles en la región en la que va a crear la CloudFormation pila. Si necesita crear un par de claves, consulte Pares de claves e instancias de Amazon EC2 en la Guía del usuario de Amazon EC2.

Lanzamiento de una Amazon DocumentDB CloudFormation pila

En esta sección se describe cómo lanzar y configurar una pila CloudFormation de Amazon DocumentDB.

1. Inicie sesión Consola de administración de AWS enhttps://console.aws.amazon.com/.
2. En la tabla siguiente se muestran las plantillas de pilas de Amazon DocumentDB para cada Región de AWS. Selecciona Launch Stack para el sitio en el que Región de AWS quieres lanzar tu pila.

Region	Ver plantilla	Ver en Designer	Lanzar
Este de EE. UU. (Ohio)	Ver plantilla	Ver en Designer	Launch Stack
Este de EE. UU. (Norte de Virginia)	Ver plantilla	Ver en Designer	Launch Stack
Oeste de EE. UU. (Oregón)	Ver plantilla	Ver en Designer	Launch Stack
Asia-Pacífico (Mumbai)	Ver plantilla	Ver en Designer	Launch Stack
Asia-Pacífico (Seúl)	Ver plantilla	Ver en Designer	Launch Stack

Region	Ver plantilla	Ver en Designer	Lanzar
Asia-Pacífico (Singapur)	Ver plantilla	Ver en Designer	Launch Stack
Asia-Pacífico (Sídney)	Ver plantilla	Ver en Designer	Launch Stack
Asia-Pacífico (Tokio)	Ver plantilla	Ver en Designer	Launch Stack
Canadá (centro)	Ver plantilla	Ver en Designer	Launch Stack
Europa (Fráncfort)	Ver plantilla	Ver en Designer	Launch Stack
Europa (Irlanda)	Ver plantilla	Ver en Designer	Launch Stack
Europa (Londres)	Ver plantilla	Ver en Designer	Launch Stack
Europa (París)	Ver plantilla	Ver en Designer	Launch Stack

3. Crear pila: describe la plantilla de Amazon DocumentDB que ha seleccionado. Cada pila se basa en una plantilla (un archivo JSON o YAML) que contiene la configuración de los AWS recursos que quieres incluir en la pila. Como eligió lanzar una pila a partir de las plantillas proporcionadas anteriormente, su plantilla ya se configuró para crear una pila de Amazon DocumentDB para la Región de AWS que eligió.

   Cuando lanza una CloudFormation pila, la protección contra la eliminación de su clúster de Amazon DocumentDB está deshabilitada de forma predeterminada. Si desea habilitar la protección de eliminación para el clúster, complete los pasos siguientes. De lo contrario, elija Next (Siguiente) para continuar con el paso siguiente.

   Para habilitar la protección de eliminación para el clúster de Amazon DocumentDB:

   1. Elija View in Designer (Ver en Designer) en la esquina inferior derecha de la página Create stack (Crear pila).

2. Modifique la plantilla con el editor JSON y YAML integrado en la página de CloudFormation diseño resultante de la consola. Desplácese hasta la sección `Resources` y modifíquela para incluir `DeletionProtection`, de la siguiente manera. Para obtener más información sobre el uso de CloudFormation Designer, consulte ¿Qué es CloudFormation Designer? .

   JSON:

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

   YAML:

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

3. Elija Create stack (Crear pila)
   ()
   en la esquina superior izquierda de la página para guardar los cambios y crear una pila con estos cambios habilitados.

4. Después de guardar los cambios, se le redirigirá a la página Create stack (Crear pila).
5. Elija Siguiente para continuar.

4. Especificar los detalles de la pila: introduce el nombre y los parámetros de la pila de su plantilla. Los parámetros se definen en la plantilla y le permiten ingresar valores personalizados al crear o actualizar una pila.
   - En Stack name (Nombre de la pila), escriba un nombre para su pila o acepte el nombre proporcionado. El nombre de la pila puede incluir letras (A-Z y a-z), números (0-9) y guiones (-).
   - En Parameters (Parámetros), escriba los siguientes detalles:
     - DBClusterName— Introduzca un nombre para su clúster de Amazon DocumentDB o acepte el nombre proporcionado.

       Restricciones en cuanto a la nomenclatura de los clústeres:
       - Debe tener [1-63] letras, números o guiones.
       - El primer carácter debe ser una letra.
       - No puede terminar con un guion ni contener dos guiones consecutivos.
       - Debe ser único para todos los clústeres de Amazon RDS, Neptune y Amazon DocumentDB por región. Cuenta de AWS
     - DBInstanceClass— En la lista desplegable, seleccione la clase de instancia para su clúster de Amazon DocumentDB.
     - DBInstanceName— Introduzca un nombre para su instancia de Amazon DocumentDB o acepte el nombre proporcionado.

       Restricciones en cuanto a la nomenclatura de las instancias:
       - Debe tener [1-63] letras, números o guiones.
       - El primer carácter debe ser una letra.
       - No puede terminar con un guion ni contener dos guiones consecutivos.
       - Debe ser único para todas las instancias de Amazon RDS, Neptune y Amazon DocumentDB por región. Cuenta de AWS
     - MasterPassword— La contraseña de la cuenta de administrador de la base de datos.
     - MasterUser— El nombre de usuario de la cuenta de administrador de la base de datos MasterUser Debe empezar por una letra y solo puede contener caracteres alfanuméricos.

Elija Next (Siguiente) para guardar los cambios y continuar.

5. Configure opciones de pila: configure las etiquetas, los permisos y las opciones adicionales de su pila.

   - Etiquetas: especifique los pares de etiquetas (clave-valor) para aplicarlos a los recursos de su pila. Puede agregar hasta 50 etiquetas únicas para cada pila.
   - Permisos: opcionales. Elija un rol de IAM para definir de forma explícita cómo CloudFormation puede crear, modificar o eliminar los recursos de la pila. Si no eliges un rol, CloudFormation usa los permisos en función de tus credenciales de usuario. Antes de especificar un rol de servicio, asegúrese de que tiene permiso para pasarlo (`iam:PassRole`). El permiso `iam:PassRole` especifica los roles que puede utilizar.

     > **Note**
     >
     > Cuando especificas un rol de servicio, CloudFormation siempre lo usa para todas las operaciones que se realizan en esa pila. Otros usuarios con permisos para realizar operaciones en esta pila podrán utilizar este rol, aunque no tengan permiso para pasarlo. Si el rol incluye permisos que el usuario no debería tener, puede escalar involuntariamente los permisos de un usuario. Asegúrese de que el rol concede privilegios mínimos.

   - Opciones avanzadas: puede definir las siguientes opciones avanzadas:

     - Política de pilas: opcional. Define los recursos que desea proteger de actualizaciones involuntarias durante una actualización de pila. De forma predeterminada, se pueden actualizar todos los recursos durante una actualización de pila.

       Puede introducir la política de pilas directamente como JSON o cargar un archivo JSON que contenga la política de pilas. Para obtener más información, consulte Evitar actualizaciones en los recursos de la pila.
     - Configuración de reversión: opcional. Especifique CloudWatch los registros y las alarmas CloudFormation para supervisarlos al crear y actualizar la pila. Si la operación supera un umbral de alarma, CloudFormation la revierte.
     - Opciones de notificación: opcional. Especifique los temas para el sistema de notificación simple (SNS).
     - Opciones de creación de la pila: opcional. Puede especificar las opciones siguientes:

- Restauración en caso de error: si la pila se debería restaurar o no en caso de error de creación de la pila.
- Tiempo de espera: se han agotado los minutos de espera de la creación de la pila.
- Protección de terminación: impide que se elimine una pila accidentalmente.

> **Note**
>
> CloudFormation la protección contra la terminación es diferente del concepto de protección contra la eliminación de Amazon DocumentDB. Para obtener más información, consulte Protección de terminación y protección de eliminación.

Elija Siguiente para continuar.

6. Revisar <stack-name>: revise la plantilla, los detalles y las opciones de configuración de la pila. También puede abrir un quick-create link (enlace de creación rápida) en la parte inferior de la página para crear pilas con las mismas configuraciones básicas que esta.

   - Elija Create (Crear) para crear la pila.
   - De forma alternativa, puede elegir Create change set (Crear conjunto de cambios). Un conjunto de cambios es una vista previa de cómo se configurará esta pila antes de crear la pila. Esto le permite examinar varias configuraciones antes de ejecutar el conjunto de cambios.

Acceso al clúster de Amazon DocumentDB

Una vez completada la CloudFormation pila, puede usar una instancia de Amazon EC2 para conectarse a su clúster de Amazon DocumentDB. Para obtener información sobre la conexión a una instancia de Amazon EC2 mediante SSH, consulte Conexión a la instancia de Linux en Guía del usuario de Amazon EC2.

Una vez conectado, consulte las siguientes secciones que contienen información sobre cómo usar Amazon DocumentDB.

- Paso 3: Insertar y consultar datos
- Administración de clústeres e instancias de Amazon DocumentDB
- Supervisión y registro en Amazon DocumentDB

Protección de terminación y protección de eliminación

Una de las mejores prácticas de Amazon DocumentDB consiste en habilitar la protección contra eliminaciones y cancelaciones. CloudFormation la protección contra la cancelación es una función claramente diferente de la función de protección contra la eliminación de Amazon DocumentDB.

- Protección contra la terminación: puede evitar que una pila se elimine accidentalmente habilitando la protección contra la terminación de su CloudFormation pila. Si un usuario intenta eliminar una pila con protección de terminación habilitada en ella, la eliminación falla y la pila no cambia. La protección de terminación está deshabilitada de forma predeterminada cuando creas una pila usando CloudFormation. Es posible habilitar la protección de terminación en una pila al crearla. Para obtener más información, consulte Configuración de las opciones de CloudFormation pila.
- Protección de eliminación: Amazon DocumentDB también ofrece la posibilidad de habilitar la protección contra la eliminación de un clúster. Si un usuario intenta eliminar un clúster de Amazon DocumentDB con la protección de eliminación habilitada en él, la eliminación falla y el clúster permanece sin cambios. La protección contra la eliminación, cuando está habilitada, protege contra las eliminaciones accidentales de Amazon DocumentDB Consola de administración de AWS AWS CLI, y. CloudFormation Para obtener más información sobre cómo habilitar y deshabilitar la protección de eliminación para un clúster de Amazon DocumentDB, consulte Protección contra eliminación.

Migración y actualización de Amazon DocumentDB

Amazon DocumentDB es una base de datos administrada que es compatible con la API de MongoDB. Puede migrar sus datos a Amazon DocumentDB desde bases de datos de MongoDB que se ejecuten en las instalaciones o en Amazon Elastic Compute Cloud (Amazon EC2) utilizando el proceso que se detalla en esta sección.

Temas

- Migre a Amazon DocumentDB mediante AWS Servicio de migración de bases de datos (DMS): guía de inicio rápido
- Manual de procedimientos de migración de Amazon DocumentDB
- Migración desde Couchbase Server
- Fechas de soporte de la versión del motor de Amazon DocumentDB
- Actualización local de la versión principal Amazon DocumentDB
- Actualización de la versión secundaria de Amazon DocumentDB
- Actualización del clúster de Amazon DocumentDB mediante AWS Database Migration Service
- Uso de una versión de soporte a largo plazo (LTS)
- Soporte extendido de Amazon DocumentDB

Migre a Amazon DocumentDB mediante AWS Servicio de migración de bases de datos (DMS): guía de inicio rápido

Temas

- Preparación del origen de DMS
- Configuración de DMS
- Habilitación de la compresión de DocumentDB
- Creación de una tarea de replicación
- Monitorear el progreso
- Información adicional

Preparación del origen de DMS

Consulte Habilitación de flujos de cambios para habilitar los flujos de cambios de DocumentDB o habilitar el oplog de MongoDB para admitir la captura de datos de cambio (CDC) de DMS.

- El origen de DMS debe retener todos los cambios en curso hasta que se complete la carga completa de DMS para todas las colecciones incluidas.
- Los flujos de cambios de DocumentDB se basan en el tiempo. Asegúrese de que la configuración de `change_stream_log_retention_duration` sea lo suficientemente grande como para cubrir el tiempo necesario para concluir la carga completa.
- El oplog de MongoDB tiene un tamaño fijo. Asegúrese de que tiene el tamaño adecuado para admitir todas las operaciones durante la carga completa.

Configuración de DMS

Cree puntos de conexión de instancia, origen y destino de DMS y pruebe cada punto de conexión.

Habilitación de la compresión de DocumentDB

Amazon DocumentDB 5.0: habilite la compresión adjuntando un grupo de parámetros personalizado a su clúster de Amazon DocumentDB y configurando el parámetro en. `default_collection_compression` `enabled` Para obtener más información, consulte Gestión de la compresión de documentos a nivel de colección.

Amazon DocumentDB 8.0: la compresión Zstd está habilitada de forma predeterminada. Para obtener más información, consulte Administración de la compresión basada en diccionarios en Amazon DocumentDB 8.0.

Creación de una tarea de replicación

1. En la consola de DMS, en el panel de navegación, elija Migrar o replicar y, a continuación, elija Tareas.
2. Seleccione Crear tarea.
3. En la página Crear tarea, en la sección Configuración de tareas:
    - Introduzca un Identificador de tarea único y significativo (por ejemplo, «mongodb-docdb-replication»).

- Elija el punto de conexión de origen que creó anteriormente en el menú desplegable de Punto de conexión de la base de datos de origen.
- Elija el punto de conexión de destino que creó anteriormente en el menú desplegable de Punto de conexión de la base de datos de destino.
- En Tipo de tarea, elija Migrar y replicar.

4. En la sección Configuración:

   - Para ver los registros de tareas, marque la casilla Activar los CloudWatch registros.
   - En Modo de edición (en la parte superior de la sección), elija Editor JSON y defina los siguientes atributos:
     - Establezca `ParallelApplyThreads` en 5 (en `TargetMetadata`). Esto permite aproximadamente 1000 operaciones de insert/update /delete por segundo en los CDC.
     - Establezca `MaxFullLoadSubTasks` en 16 (en `FullLoadSettings`). Considere aumentar el valor en función del tamaño de la instancia.
     - Para colecciones grandes (más de 100 GB), habilite la partición automática (en Correspondencia de tablas y en el atributo `parallel-load`):
       - «type»: «partitions-auto»
       - «number-of-partitions»: 16

Monitorear el progreso

Use la AWS DMS consola o cree un panel personalizado (herramienta de panel de control) para realizar un seguimiento de la migración. Debe concentrarse en las siguientes métricas:

- FullLoadThroughputBandwidthTarget— Mide el ancho de banda de red (en KB/second pulgadas) utilizado por el DMS al transferir datos a la base de datos de destino durante la fase de carga completa de la migración.
- CDCLatencyTarget— Mide el tiempo de demora (en segundos) entre el momento en que se produce un cambio en la base de datos de origen y el momento en que ese cambio se aplica a la base de datos de destino.
- CDCThroughputRowsTarget— Mide el número de filas por segundo que DMS aplica a la base de datos de destino durante la fase de replicación en curso de la migración.

Información adicional

Para obtener más información sobre Amazon DocumentDB y consulte AWS DMS:

- Manual de procedimientos de migración de Amazon DocumentDB
- Migración desde MongoDB a Amazon DocumentDB

Manual de procedimientos de migración de Amazon DocumentDB

Este manual proporciona una guía completa para migrar una base de datos de MongoDB a Amazon DocumentDB mediante (DMS). AWS Database Migration Service Está diseñado para ayudar a los administradores de bases de datos, ingenieros de nube y desarrolladores durante todo el proceso de migración, desde la detección inicial hasta la validación posterior a la migración.

Dadas las diferencias en la implementación y las características compatibles entre MongoDB y Amazon DocumentDB, en este manual de procedimientos se hace hincapié en un enfoque estructurado y sistemático. Se describen las evaluaciones esenciales previas a la migración, se destacan las consideraciones de compatibilidad y se detallan las tareas clave necesarias para garantizar una migración correcta con un mínimo de interrupciones.

El manual de procedimientos se organiza en los siguientes temas:

- Compatibilidad: conozca las características y los tipos de datos de MongoDB compatibles con Amazon DocumentDB e identifique posibles incompatibilidades.
- Detección de la carga de trabajo— Analice las cargas de trabajo de MongoDB existentes, incluidos read/write los patrones, los volúmenes de datos y las líneas base de rendimiento.
- Migración de índices: analice las estrategias para extraer y transformar los índices de MongoDB para obtener un rendimiento óptimo en Amazon DocumentDB.
- Migración de usuario: detalle el enfoque para migrar los usuarios, los roles y los controles de acceso de las bases de datos a Amazon DocumentDB.
- Migración de datos— Cubra varios métodos de migración de datos AWS DMS, incluida la captura de datos con carga completa y cambios (CDC).
- Supervisión: detalle varios enfoques de supervisión cuando haga una migración mediante DMS o herramientas nativas.
- Validación: proporcione procedimientos para las comprobaciones de integridad de los datos, la validación funcional y la comparación del rendimiento después de la migración.

Si se siguen las instrucciones de este manual de procedimientos, los equipos pueden garantizar una transición fluida, segura y eficiente a Amazon DocumentDB, a la vez que conservan la funcionalidad de la aplicación y minimizan los riesgos.

Compatibilidad

Temas

- Compatibilidad de características básicas
- Herramienta de evaluación de compatibilidad de Amazon DocumentDB

Cuando se realiza una migración de MongoDB a Amazon DocumentDB, es esencial realizar una evaluación inicial exhaustiva y comprobar la compatibilidad de las características para que la migración se lleve a cabo correctamente. Este proceso comienza con un inventario completo de las características de MongoDB, incluidos los operadores de canalización de agregación, los patrones de consulta, los índices y los modelos de datos.

Dado que Amazon DocumentDB es compatible con las API 3.6, 4.0, 5.0 y 8.0 de MongoDB, las aplicaciones que utilizan MongoDB-specific funciones más recientes pueden requerir una refactorización. Las áreas críticas que se deben evaluar incluyen los mecanismos de partición (Amazon DocumentDB utiliza un enfoque diferente), las implementaciones de transacciones, la funcionalidad de los flujos de cambios y los tipos de índices (especialmente los índices dispersos y parciales).

Las características de rendimiento también difieren, ya que Amazon DocumentDB está optimizada para cargas de trabajo empresariales con un rendimiento predecible. Las pruebas deben incluir la ejecución de cargas de trabajo representativas en ambos sistemas para identificar los patrones de consulta que puedan requerir optimización.

Durante la fase de evaluación, es importante supervisar los planes de ejecución para detectar posibles brechas de rendimiento. Esto ayuda a crear una hoja de ruta de migración clara, ya que identifica los cambios necesarios en las aplicaciones y establece plazos realistas para una transición fluida.

Compatibilidad de características básicas

Soporte de características integral

- Operaciones CRUD: disfrute de una compatibilidad total con todas las operaciones básicas de creación, lectura, actualización y eliminación, incluidos los operadores masivos y de consulta, lo que proporciona una compatibilidad perfecta entre aplicaciones.
- Amplias capacidades de indexación: aproveche el soporte integral para índices de un solo campo, compuestos, TTL, parciales, dispersos y de 2dsphere para optimizar el rendimiento de sus consultas y los índices de texto (versión 5) para las búsquedas basadas en texto.
- Enterprise-grade replicación: benefíciese de un sólido mecanismo de conmutación por error automático con réplicas de lectura para lograr una alta disponibilidad sin sobrecarga operativa.
- Soluciones de respaldo avanzadas: descanse tranquilo con un sistema de respaldo automatizado que incluye Point-in-Time recuperación (PITR) e instantáneas manuales bajo demanda para proteger los datos.

Mejorado AWS-funciones integradas

- Agregación simplificada: aproveche las etapas de agregación más utilizadas (`$match`, `$group`, `$sort`, `$project`, etc.) con un rendimiento optimizado para las cargas de trabajo empresariales.
- Soporte de transacciones: implemente transacciones con documentos y colecciones múltiples, lo que resulta perfecto para la mayoría de las necesidades de las aplicaciones empresariales.
- Real-time seguimiento de datos: habilite los flujos de cambios con un simple comando y aumente el período de retención de los flujos de cambios mediante una sencilla configuración de grupos de parámetros para monitorear los cambios de datos en tiempo real.
- Location-based servicios: implemente aplicaciones geoespaciales compatibles con los índices de `$geoNear` operador y 2dsphere.
- Capacidades de búsqueda de texto: utilice la funcionalidad integrada de búsqueda de texto para las necesidades de detección de contenido.

Ventajas de la arquitectura moderna

- Cloud-native diseño: disfrute de una arquitectura AWS optimizada que reemplaza las funciones antiguas, por ejemplo, por operaciones de canalización de MapReduce agregación más eficientes.
- Seguridad mejorada: aproveche las ventajas de AWS Identity and Access Management (IAM) SCRAM-SHA-1 SCRAM-SHA-256, la autenticación mediante X.509 certificados y la autenticación basada en contraseñas.

- Rendimiento predecible: disfrute de un rendimiento uniforme optimizado específicamente para las cargas de trabajo empresariales.

Para obtener una descripción general completa de las capacidades de Amazon DocumentDB, consulte API, operaciones y tipos de datos de MongoDB admitidos en Amazon DocumentDB y Diferencias funcionales: Amazon DocumentDB y MongoDB para maximizar el potencial de su base de datos.

Amazon DocumentDB no es compatible con todos los índices que ofrece MongoDB. Ofrecemos una herramienta de indexación gratuita para comprobar la compatibilidad GitHub . Ejecute la herramienta de indexación para evaluar la incompatibilidad y planificar las soluciones en consecuencia.

Herramienta de evaluación de compatibilidad de Amazon DocumentDB

La herramienta de compatibilidad de Amazon DocumentDB GitHub analiza su MongoDB carga de trabajo e informa sobre los operadores, comandos y funciones que no son compatibles o que requieren cambios en Amazon DocumentDB. Es compatible con las versiones de destino de la 3.6 a la 8.0.1 (predeterminada: 8.0.1).

Recomendado: modo URI (5.0+) MongoDB

El modo URI se conecta directamente a la instancia de MongoDB, toma muestras de las operaciones en tiempo real y no requiere la creación de perfiles ni el acceso a los registros.

Métodos alternativos

- Log-based análisis: analiza MongoDB los registros del generador de perfiles. Captura el comportamiento real del tiempo de ejecución, pero requiere que la creación de perfiles esté habilitada y solo cubre el período de registro.
- Análisis del código fuente: escanea el código fuente de la aplicación para determinar el uso de MongoDB la API. Proporciona una cobertura completa, pero puede marcar las rutas de código no utilizadas y no puede detectar consultas creadas de forma dinámica.

Requisitos previos

- PythonMás de 3.7
- Clona la herramienta: `git clone https://github.com/awslabs/amazon-documentdb-tools.git`

- Instale las dependencias: `pip install -r compat-tool/requirements.txt`

Para obtener todos los detalles de uso y todas las opciones disponibles, consulte el archivo README en. GitHub

Detección de la carga de trabajo

La migración de MongoDB a Amazon DocumentDB requiere un conocimiento profundo de la carga de trabajo de la base de datos existente. La detección de la carga de trabajo es el proceso de analizar los patrones de uso de la base de datos, las estructuras de datos, el rendimiento de las consultas y las dependencias operativas para garantizar una transición fluida con una interrupción mínima. En esta sección se describen los pasos clave que implica la detección de cargas de trabajo para facilitar una migración eficaz de MongoDB a Amazon DocumentDB.

Temas

- Evaluación de la implementación de MongoDB existente
- Identificación de diferencias en los modelos de datos
- Análisis de consultas y rendimiento
- Revisión de seguridad y control de acceso
- Consideraciones operativas y de supervisión

Evaluación de la implementación de MongoDB existente

Antes de la migración, es fundamental evaluar el entorno actual de MongoDB, que incluye:

- Arquitectura del clúster: identifique la cantidad de nodos, los conjuntos de réplicas y las configuraciones de partición. Cuando migra de MongoDB a Amazon DocumentDB, es importante comprender la configuración de partición de MongoDB, ya que Amazon DocumentDB no admite la partición controlada por el usuario. Las aplicaciones diseñadas para un entorno de MongoDB con partición necesitarán cambios en la arquitectura, ya que Amazon DocumentDB utiliza un enfoque de escalado diferente con su arquitectura basada en el almacenamiento. Tendrá que adaptar su estrategia de distribución de datos y, posiblemente, consolidar las colecciones con partición cuando pase a Amazon DocumentDB.
- Almacenamiento y volumen de datos: mida el tamaño total de los datos y el tamaño del índice de su clúster. Complemente esto con la herramienta de revisión del oplog para comprender los

patrones de escritura y la velocidad de crecimiento de los datos. Para obtener más información acerca del dimensionamiento de su clúster, consulte Determinación del tamaño de las instancias.

- Patrones de carga de trabajo: analice el rendimiento de lectura y escritura, la frecuencia de ejecución de las consultas y la eficiencia de la indexación.
- Dependencias operativas: documente todas las aplicaciones, los servicios y las integraciones que se basan en MongoDB.

Identificación de diferencias en los modelos de datos

Aunque Amazon DocumentDB sí lo es MongoDB-compatible, existen diferencias en cuanto a las funciones compatibles, como:

- Transacciones: Amazon DocumentDB admite transacciones ACID, pero con algunas Limitaciones.
- Diseño de esquemas: asegúrese de que las estructuras de los documentos, los documentos incrustados y las referencias se ajusten a las prácticas recomendadas de Amazon DocumentDB.

Análisis de consultas y rendimiento

Comprender el comportamiento de las consultas ayuda a optimizar el rendimiento de la migración y después de la migración. Las siguientes son áreas clave que se deben analizar:

- Consultas lentas: identifique las consultas con un tiempo de ejecución elevado mediante las herramientas de creación de perfiles de MongoDB.
- Patrones de consulta: clasifique los tipos de consultas más frecuentes, incluidas las operaciones CRUD y las agregaciones.
- Uso de índices: evalúe si los índices se utilizan de forma eficaz o si es necesario optimizarlos en Amazon DocumentDB. Para evaluar el uso de los índices y optimizar el rendimiento en Amazon DocumentDB, use la etapa de canalización de agregación de `$indexStats` en combinación con el método de `explain()` en sus consultas críticas. Comience por ejecutar `db.collection.aggregate([{$indexStats{}}])` para identificar qué índices se están utilizando. Puede realizar un análisis más detallado ejecutando las consultas más frecuentes con `explainPlan`.
- Concurrencia y distribución de la carga de trabajo: evalúe las relaciones de lectura y escritura, la agrupación de conexiones y los cuellos de botella en el rendimiento.

Revisión de seguridad y control de acceso

Autenticación y autorización

- RBAC de MongoDB a IAM y RBAC de Amazon DocumentDB: asigne los usuarios y funciones del control de acceso basado en roles de MongoDB a las políticas AWS Identity and Access Management (IAM) y a los usuarios de autenticación SCRAM de Amazon DocumentDB.
- Estrategia de migración de usuarios: planifique la migración de los usuarios de bases de datos, los roles personalizados y los privilegios a los mecanismos de autenticación compatibles con Amazon DocumentDB.
- Diferencias de privilegios: identifique los privilegios de MongoDB sin equivalentes directos de Amazon DocumentDB (por ejemplo, roles de administración de clústeres).
- Autenticación de aplicaciones: actualice las cadenas de conexión y la administración de credenciales para las políticas de contraseñas de Amazon DocumentDB. Puede usar el administrador de secretos para almacenar sus credenciales y rotar las contraseñas.
- Administración de cuentas de servicio: establezca procesos para administrar las credenciales de las cuentas de servicio en. AWS Secrets Manager
- Implementación de privilegios mínimos: revise y perfeccione los controles de acceso para implementar los principios de privilegios mínimos en el nuevo entorno.

Cifrado

Asegúrese de que el cifrado en reposo y en tránsito se ajuste a los requisitos de conformidad.

Configuración de red

Planifique la configuración de la nube privada virtual (VPC) y las reglas de los grupos de seguridad.

Consideraciones operativas y de supervisión

Para mantener la fiabilidad del sistema, la detección de la carga de trabajo también debe incluir lo siguiente:

- Estrategia de copia de seguridad y restauración: evalúe los métodos de copia de seguridad existentes y las capacidades de copia de seguridad de Amazon DocumentDB.
- AWS Backup integración: aproveche la administración centralizada AWS Backup de copias de seguridad en todos AWS los servicios, incluido Amazon DocumentDB.

- CloudWatch métricas: asigne las métricas de monitoreo de MongoDB a las métricas de Amazon DocumentDB CloudWatch para la CPU, la memoria, las conexiones y el almacenamiento.
- Información sobre rendimiento: implemente la Información de rendimiento de Amazon DocumentDB para visualizar la carga de la base de datos y analizar los problemas de rendimiento con análisis de consultas detallados.
- Generador de perfiles: configure el generador de perfiles de Amazon DocumentDB para capturar las operaciones de ejecución lenta (similar al generador de perfiles de MongoDB, pero con la configuración de Amazon). DocumentDB-specific
  - Realice la habilitación mediante grupos de parámetros con los umbrales adecuados.
  - Analice los datos del generador de perfiles para identificar oportunidades de optimización
- CloudWatch Eventos: configure la supervisión basada en eventos para los eventos del clúster de Amazon DocumentDB.
  - Configure las notificaciones para los eventos de copia de seguridad, los períodos de mantenimiento y las conmutaciones por error.
  - Intégrelo con Amazon SNS para recibir alertas y respuestas automatizadas. AWS Lambda
- Registro de auditoría: planifique la configuración del registro de auditoría para rastrear la actividad de los usuarios y los eventos relevantes para la seguridad.
- Monitoreo mejorado: habilite el monitoreo mejorado de OS-level métricas granulares a intervalos de 1 segundo.

Cree un nuevo clúster de destino

Cree un clúster de Amazon DocumentDB siguiendo los pasos descritos enCreación de un clúster de Amazon DocumentDB. Al configurar el clúster para la migración, aplique estas recomendaciones:

- Clase de instancia: elija R8G para obtener el mejor rendimiento. Si el R8G no está disponible en la región de destino, utilice el R6G. Elija una instancia lo más grande posible para obtener el mejor rendimiento a plena carga. Reduzca la escala una vez finalizada la migración.
- Versión del motor: más reciente (8.0.1)
- Cantidad de instancias: elija 1 instancia para minimizar los costos durante la migración. Escale a 3 instancias para obtener una alta disponibilidad una vez finalizada la migración de carga completa.
- Configuración de red: asegúrese de que su grupo de seguridad de Amazon DocumentDB permita las conexiones entrantes desde el grupo de seguridad de la instancia de replicación del DMS (para

las migraciones en línea) o desde el host de migración de EC2 (para las migraciones sin conexión mediante mongorestore).

Migración de índices

La migración de MongoDB a Amazon DocumentDB implica transferir no solo datos, sino también índices para mantener el rendimiento de las consultas y optimizar las operaciones de la base de datos. En esta sección, se describe el proceso paso a paso detallado para la migración de índices de MongoDB a Amazon DocumentDB y, al mismo tiempo, se garantiza la compatibilidad y la eficacia.

Uso de la herramienta de indexación de Amazon DocumentDB

Clona la herramienta de indexación en https://github.com/awslabs/amazon-documentdb-tools/blob/master/index-tool/README.md GitHub

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

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

Exportación de índices desde MongoDB (si se migra desde MongoDB)

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

Verificación de índices

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

Índices de importación

```
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'
```

Migración de usuario

La migración de usuarios de MongoDB a Amazon DocumentDB es esencial para mantener el control de acceso, la autenticación y la seguridad de la base de datos. En esta sección se describen los pasos detallados para migrar correctamente los usuarios de MongoDB y, al mismo tiempo, conservar sus roles y permisos mediante la herramienta de exportación de usuarios de Amazon DocumentDB.

Uso de la herramienta de exportación de usuarios de Amazon DocumentDB

`Export Users tool`Exporta los usuarios y roles de MongoDB o Amazon DocumentDB a JavaScript archivos, que luego se pueden usar para recrearlos en otro clúster.

Requisitos previos

```
# 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
```

Paso 1: exportar usuarios y roles

```
# 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"
```

Paso 2: editar el archivo de usuarios

Las contraseñas no se exportan por motivos de seguridad. Abra el `mongodb-users.js` archivo y añada una contraseña para cada usuario sustituyendo el pwd valor de cada `createUser` instrucción:

```
db.getSiblingDB("admin").createUser({
```

```
  user: "appuser",
  pwd: "REPLACE_THIS_PASS",
  roles: [
    { role: "readWrite", db: "mydb" }
  ]
})
```

Paso 3: restaurar roles personalizados en Amazon DocumentDB

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

Paso 4: restaurar usuarios en 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
```

Notas importantes

- Las contraseñas no se exportan por motivos de seguridad y deben añadirse manualmente al archivo users.js.
- Los roles se deben importar antes que los usuarios para garantizar una asignación de roles adecuada.
- La herramienta genera JavaScript archivos que se pueden ejecutar directamente con el shell mongosh.
- Los roles personalizados y sus privilegios se conservan durante la migración.
- Este enfoque permite revisar y modificar los permisos de los usuarios antes de importarlos.

Este método proporciona un enfoque seguro y flexible para migrar usuarios y roles de MongoDB a Amazon DocumentDB y, al mismo tiempo, permite restablecer las contraseñas durante el proceso de migración.

Migración de datos

Temas

- Migración en línea
- Migración sin conexión

Migración en línea

En esta sección se proporcionan los pasos detallados para realizar una migración en línea de MongoDB a Amazon DocumentDB con el fin de reducir al mínimo el tiempo de inactividad y AWS DMS lograr una replicación continua. Para empezar, debe configurar un clúster de Amazon DocumentDB como destino y asegurarse de que su instancia de MongoDB esté correctamente configurada como el origen, lo que normalmente requiere el modo de conjunto de réplicas para la captura de datos de cambios. A continuación, debe crear una instancia de replicación de DMS y definir los puntos de conexión de origen y destino con los detalles de conexión necesarios. Tras validar los puntos de conexión, debe configurar e iniciar una tarea de migración que puede incluir la carga completa de los datos, la replicación continua o ambas.

Cree un nuevo clúster de destino

Consulte Cree un nuevo clúster de destino.

Configurar el origen

Tanto MongoDB como Amazon DocumentDB pueden servir como orígenes de migración, según la situación:

- MongoDB como fuente: es común cuando se migra de un MongoDB local o autogestionado a Amazon DocumentDB u otros servicios de bases de datos. AWS Requiere ejecutarse en modo conjunto de réplicas con un oplog del tamaño adecuado (asegúrese de que tenga el tamaño adecuado para contener todas las operaciones durante la carga completa) para permitir la captura de datos de cambios durante la migración.
- Amazon DocumentDB como fuente: normalmente se usa para la replicación entre regiones, las actualizaciones de versiones o la migración a otros servicios de bases de datos como MongoDB

Atlas. Se requiere Habilitación de flujos de cambios mediante la configuración del parámetro `change_stream_log_retention_duration` en el grupo de parámetros del clúster para capturar los cambios en curso durante la migración. Asegúrese de que la configuración de la `change_stream_log_retention_duration` sea lo suficientemente prolongada como para cubrir el tiempo necesario para concluir la carga completa.

Antes de iniciar la migración, configure su fuente para permitir el acceso. AWS DMS

Cree un usuario de MongoDB con los permisos adecuados:

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

Configure la red y la autenticación.

Cuando configure la conectividad de red para la migración de MongoDB a DMS:

EC2-hosted MongoDB como origen

- Modifique el grupo de seguridad de EC2 para permitir el tráfico entrante desde el grupo de seguridad de la instancia de replicación de DMS.
- Agregue una regla para el puerto TCP 27017 (o su puerto MongoDB personalizado).
- Utilice el identificador del grupo de seguridad de la instancia de replicación de DMS como origen para un control de acceso preciso.
- Asegúrese de que la subred de la instancia de EC2 tenga una ruta hacia la subred de la instancia de replicación de DMS.

On-premises MongoDB como origen

- Configure su firewall para permitir las conexiones entrantes desde las direcciones IP públicas de la instancia de replicación de DMS.
- Si utiliza una VPN Direct Connect o una VPN, asegúrese de que el enrutamiento entre la red y la VPC que contiene la instancia de DMS.
- Pruebe la conectividad mediante comandos telnet o nc desde la subred de DMS a su servidor MongoDB.

MongoDB Atlas como origen

- Agregue las direcciones IP de una instancia de replicación de DMS a la lista de IP permitidas de MongoDB Atlas.
- Configure la interconexión de VPC entre la AWS VPC y MongoDB Atlas VPC si Atlas se está ejecutando en ella. AWS
- Configure la AWS PrivateLink conectividad privada (nivel empresarial), si se ejecuta en otro proveedor de nube.
- Cree un usuario dedicado con read/write los permisos adecuados.
- Utilice una cadena de conexión de MongoDB Atlas con el modo SSL establecido en «verify-full».
- Asegúrese de que el tamaño del oplog sea suficiente para la duración de la migración.

Amazon DocumentDB como origen

Configure el grupo de seguridad de Amazon DocumentDB de origen para permitir el tráfico entrante desde el grupo de seguridad de la instancia de replicación de DMS.

Creación de una instancia de replicación de DMS

Use DMS Buddy on GitHub para crear una infraestructura de migración óptima con configuraciones de DMS y tamaños de instancia óptimos. Si prefiere realizar la configuración manualmente, siga estos pasos:

1. En la AWS DMS consola, en el panel de navegación, elija Migrar o replicar y, a continuación, elija Instancias aprovisionadas.
2. Introduzca los detalles de la instancia de replicación:
   - Nombre de la instancia: elija un nombre único.
   - Clase de instancia: selecciónela en función de la carga de trabajo. Ejemplo: dms.r7i.large (cargas de trabajo pequeñas), dms.r7i.4xlarge (cargas de trabajo grandes).
   - Versión del motor: 3.5.4
   - Almacenamiento asignado: el valor predeterminado es de 50 GB (aumente si es necesario). Esto viene determinado por el tamaño del documento updates/second y la duración total de la carga.
   - Multi-AZ Implementación: habilite la alta disponibilidad, si es necesario.
   - Elija la misma VPC que Amazon DocumentDB.

- Asegúrese de que los grupos de seguridad permitan el tráfico entrante desde el origen y Amazon DocumentDB.

3. Elija Crear instancia de replicación y espere a que el estado esté disponible.

Creación de puntos de conexión de DMS

Creación de un punto de conexión de origen

Si se toma MongoDB como origen

1. En la consola de DMS, en el panel de navegación, elija Migrar o replicar y, a continuación, elija Puntos de conexión.
2. Seleccione Crear punto de conexión.
3. En la página Crear punto de conexión, elija Punto de conexión de origen.
4. En la sección Configuración de punto de conexión:
   - Introduzca un Identificador de punto de conexión único y significativo (por ejemplo, «mongodb-source»).
   - Elija MongoDB como motor de origen.
   - Para acceder a la base de datos de puntos de conexión, elija Proporcionar información de acceso manualmente.
   - En Nombre del servidor, introduzca su*MongoDB server DNS name/IP address*.
   - En Puerto, introduzca 27017 (puerto MongoDB predeterminado).
   - Para el modo de autenticación, elija el modo apropiado para su aplicación (password/SSL) (el predeterminado es el administrador de secretos).
   - Si el Modo de autenticación es Contraseña, proporcione:
     - Nombre de usuario y Contraseña: escriba las credenciales de MongoDB.
     - Nombre de base de datos: el nombre de la base de datos de origen.
     - Mecanismo de autenticación: SCRAM-SHA-1 (predeterminado) o mecanismo apropiado
5. En Modo de metadatos, deje la configuración predeterminada del documento.
6. Atributos de conexión adicionales:
   - authSource=admin (si la base de datos de autenticación es diferente).
   - replicaSet=<your-replica-set-name> (obligatorio para la CDC).

Si se toma Amazon DocumentDB como origen

1. En la consola de DMS, en el panel de navegación, elija Migrar o replicar y, a continuación, elija Puntos de conexión.
2. Seleccione Crear punto de conexión.
3. En la página Crear punto de conexión, elija Punto de conexión de origen.
4. En la sección Configuración de punto de conexión:
   - Introduzca un Identificador de punto de conexión único y significativo (por ejemplo, «docdb-source»).
   - Elija Amazon DocumentDB como Motor de origen.
   - Para acceder a la base de datos de puntos de conexión, elija Proporcionar información de acceso manualmente.
   - Para el nombre del servidor, introduzca su*source Amazon DocumentDB cluster endpoint*.
   - En Puerto ingrese 27017 (puerto predeterminado de Amazon DocumentDB).
   - En Modo SSL, elija verify-full (recomendado para Amazon DocumentDB).
   - En Certificado CA, elija el certificado CA de Amazon RDS.
   - Para el modo de autenticación, elija el modo apropiado para su aplicación (password/SSL) (el predeterminado es el administrador de secretos).
   - Si el Modo de autenticación es Contraseña, proporcione:
     - Nombre de usuario y contraseña: introduzca las credenciales de Amazon DocumentDB.
     - Nombre de base de datos: el nombre de la base de datos de origen.
     - Mecanismo de autenticación: SCRAM-SHA-1 (predeterminado) o mecanismo apropiado
5. En Modo de metadatos, deje la configuración predeterminada del documento.

Creación de un punto de conexión de destino (Amazon DocumentDB)

1. En la consola de DMS, en el panel de navegación, elija Migrar o replicar y, a continuación, elija Puntos de conexión.
2. Seleccione Crear punto de conexión.
3. En la página Crear punto de conexión, elija Punto de conexión de destino.
4. En la sección Configuración de punto de conexión:

- Introduzca un Identificador de punto de conexión único y significativo (por ejemplo, «docdb-target»).
- Elija Amazon DocumentDB como Motor de destino.
- En Acceder a la base de datos de puntos de conexión, elija el método que desee utilizar para autenticar el acceso a la base de datos:
  - Si elige Secrets Manager de AWS , elija el secreto en el que guarda sus credenciales de Amazon DocumentDB en el campo Secreto.
  - Si elige Proporcionar información de acceso manualmente:
    - Para el nombre del servidor, introduzca su*target Amazon DocumentDB cluster endpoint*.
    - En Puerto ingrese 27017 (puerto predeterminado de Amazon DocumentDB).
    - En Modo SSL, elija verify-full (recomendado para Amazon DocumentDB).
    - En Certificado CA, descargue y especifique el paquete de certificados CA para la verificación de SSL.
    - Para el modo de autenticación, elija el modo apropiado para su aplicación (password/SSL) (el predeterminado es el administrador de secretos).
    - Si el Modo de autenticación es Contraseña, proporcione:
      - Nombre de usuario y contraseña: introduzca las credenciales de Amazon DocumentDB.
      - Nombre de base de datos: el nombre de la base de datos de origen.
      - Mecanismo de autenticación: SCRAM-SHA-1 (predeterminado) o mecanismo apropiado

5. En Modo de metadatos, deje la configuración predeterminada del documento.

Creación de una tarea de replicación

1. En la consola de DMS, en el panel de navegación, elija Migrar o replicar y, a continuación, elija Tareas.
2. Seleccione Crear tarea.
3. En la página Crear tarea, en la sección Configuración de tareas:

   - Introduzca un Identificador de tarea único y significativo (por ejemplo, «mongodb-docdb-replication»).
   - Elija el punto de conexión de origen que creó anteriormente en el menú desplegable de Punto de conexión de la base de datos de origen.

- Elija el punto de conexión de destino que creó anteriormente en el menú desplegable de Punto de conexión de la base de datos de destino.
- Elija la instancia de replicación que creó anteriormente en el menú desplegable de la instancia aprovisionada.
- En Tipo de tarea, elija Migrar y replicar.
- ¿Durante cuánto tiempo piensa continuar con la replicación desde el origen hasta el destino? , seleccione Por tiempo limitado.

4. En la sección Configuración:

   - Para el modo Target table prepare, seleccione No hacer nada. De lo contrario, se eliminarán las colecciones e índices que ya se han creado.
   - Para los registros de tareas, marca la casilla Activar CloudWatch los registros.
   - Deja los valores predeterminados para todos los demás ajustes.

5. Volviendo a la parte superior de la sección Configuración, en el modo de edición, selecciona el editor JSON y establece los siguientes atributos:

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

   `MaxFullLoadSubTasks`Configúralo en función del recuento de recopilaciones y la capacidad de instancias. Usa valores más bajos (2 a 4) para colecciones más pequeñas o más pequeñas, y valores más altos (8 a 16) para muchas colecciones grandes. Para ver el tamaño automatizado, consulta DMS Buddy en. https://github.com/awslabs/amazon-documentdb-tools/tree/master/migration/dms_buddy GitHub

6. En la sección Mapeos de tabla, añada una nueva regla de selección.

   - En Nombre del esquema, añada la base de datos de origen que quiere migrar. Utilice % para especificar bases de datos múltiples.
   - En Tabla del esquema, añada la colección de origen que quiere migrar. Utilice % para especificar colecciones múltiples.

- En Acción, deje Incluir, la configuración predeterminada.

7. Para colecciones grandes (más de 100 GB), añada la Regla de configuración de tablas:

   - En Nombre del esquema, añada la base de datos de origen que quiere migrar. Utilice % para especificar bases de datos múltiples.
   - En Tabla del esquema, añada la colección de origen que quiere migrar. Utilice % para especificar colecciones múltiples.
   - Para el número de particiones, configúralo en función de tu carga de trabajo. Para un número reducido de colecciones grandes, defina particiones ≥ `MaxFullLoadSubTasks` para mantener ocupados a todos los trabajadores. Para muchas colecciones que se cargan en paralelo, defina las particiones en ≤, `MaxFullLoadSubTasks` ya que el recuento de colecciones en sí mismo proporciona suficiente paralelismo.

8. En la sección Evaluación previa a la migración, asegúrese de que esté desactivada. La evaluación previa a la migración no es compatible con Amazon DocumentDB como objetivo y debe estar deshabilitada.

Migración sin conexión

En esta sección se describe el proceso para realizar una migración sin conexión desde una instancia de MongoDB autogestionada a Amazon DocumentDB mediante herramientas nativas de MongoDB: `mongodump` y `mongorestore`.

Requisitos previos

Requisitos de MongoDB como origen

- Acceda a la instancia de MongoDB de origen con los permisos adecuados.
- Instale `mongodump` si es necesario (se instala durante una instalación de MongoDB).
- Asegúrese de que haya suficiente espacio en disco para los archivos de volcado.

Requisitos de Amazon DocumentDB como destino

- Asegúrese de tener aprovisionado un clúster de Amazon DocumentDB.
- Asegúrese de que haya una instancia de EC2 en la misma VPC que Amazon DocumentDB para facilitar la migración.
- La conectividad de red debe estar disponible entre el entorno de origen y Amazon DocumentDB.

- mongorestore debe estar instalado en la instancia de EC2 de migración.
- Se deben configurar los permisos de IAM adecuados para acceder a Amazon DocumentDB.

Requisitos generales

- AWS CLI debe configurarse (si usa AWS servicios para almacenamiento intermedio)
- Debe haber un ancho de banda suficiente para la transferencia de datos.
- Se debe aprobar el tiempo de inactividad (si realiza una migración en vivo, considere otros enfoques).

Preparación de un clúster de Amazon DocumentDB

Consulte Cree un nuevo clúster de destino.

Realización de volcado de datos (mongodump)

Elija una de las siguientes opciones para crear un archivo de volcado:

- Opción 1: básica

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

- Opción 2: mejor control y rendimiento

  Comprime la salida con gzip, descarga las colecciones en paralelo, habilita el TLS y lee desde una réplica secundaria, si está disponible:

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

- Opción 3: bases de datos de gran tamaño

Vacia una base de datos y una colección específicas, filtra los documentos por consulta y los envía a un único archivo comprimido:

```
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
```

Transferencia de archivos de volcado al entorno de restauración

Elija un método adecuado en función del tamaño del volcado:

- Pequeño: copie los archivos directamente en su máquina de migración (instancia de EC2 que creó anteriormente):

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

- Mediano: utilice Amazon S3 como almacenamiento intermedio:

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

- Grande: para bases de datos muy grandes, considere la posibilidad de AWS DataSync realizar una transferencia física.

Restauración de datos en Amazon DocumentDB (mongorestore)

Antes de iniciar el proceso de restauración, cree los índices en Amazon DocumentDB. Puede utilizar la herramienta de indexación de Amazon DocumentDB para exportar e importar índices.

Seleccione una de las siguientes opciones para restaurar los datos:

- Opción 1: restauración básica

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

- Opción 2: mejor control y rendimiento

  Restaura los volcados comprimidos con gzip mediante la recopilación e inserción de documentos en paralelo. Omite la restauración de índices, ya que los índices están precreados:

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

- Opción 3: bases de datos grandes o controles específicos

  Restaura desde un archivo comprimido con filtrado de espacios de nombres para incluir o excluir bases de datos y colecciones específicas. Omite la restauración de índices, ya que los índices están precreados:

```
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
```

Supervisión

En esta sección se proporciona un proceso de supervisión detallado para realizar un seguimiento del progreso, el rendimiento y el estado de una migración en curso.

Los pasos de supervisión se aplican independientemente del método de migración (AWS DMS u otras herramientas). mongodump/mongorestore

AWS DMS Supervisión de la migración (si corresponde)

Supervise las siguientes CloudWatch métricas clave:

Métricas de la fase de carga completa

- FullLoadThroughputBandwidthTarget— Ancho de banda de red (KB/second) durante la carga completa
- FullLoadThroughputRowsTarget— Número de rows/documents cargas por segundo
- FullLoadThroughputTablesTarget— Número de tables/collections finalizados por minuto
- FullLoadProgressPercent— Porcentaje de carga completa completada
- TablesLoaded— Número de cargadas tables/collections correctamente
- TablesLoading— Número de personas que se tables/collections están cargando actualmente
- TablesQueued— Número de personas en tables/collections espera de cargarse
- TablesErrored— Número de los tables/collections que no se pudieron cargar

Métricas de la fase de CDC

- CDCLatencyTarget— Tiempo de retraso (segundos) entre el cambio de fuente y la aplicación de destino
- CDCLatencySource— Tiempo de retraso (segundos) entre el cambio de fuente y la lectura del DMS
- CDCThroughputRowsTarget— Filas por segundo aplicadas durante la replicación en curso
- CDCThroughputBandwidthTarget— Ancho de banda de red (KB/second) durante los CDC
- CDCIncomingChanges— Número de eventos de cambio recibidos de la fuente
- CDCChangesMemoryTarget— Memoria utilizada (MB) para almacenar los cambios en el lado objetivo

Métricas de recursos

- CPUUtilization: uso de la CPU de la instancia de replicación.
- FreeableMemory— Memoria disponible en la instancia de replicación
- FreeStorageSpace— Almacenamiento de información disponible en la instancia de replicación
- NetworkTransmitThroughput— Rendimiento de red para la instancia de replicación
- NetworkReceiveThroughput— Rendimiento de red para la instancia de replicación

Métricas de errores

- ErrorsCount— Número total de errores durante la migración
- TableErrorsCount— Número de errores específicos de la tabla
- RecordsErrorsCount— Número de errores específicos de un registro

Cree CloudWatch alarmas para métricas críticas, como `CDCLatencyTarget` por ejemplo, `CPUUtilization` para recibir notificaciones si el rendimiento de la migración se deteriora.

Registros de DMS (registros) CloudWatch

1. Ve a la consola de Amazon CloudWatch Logs.
2. Busque y elija un elemento de su grupo de registro. Tendrá un aspecto similar a «dms-tasks –».
3. Busque flujos de registro que puedan contener información de error:
   - Flujos con la palabra «error» en el nombre
   - Flujos con identificadores de tareas o nombres de puntos de conexión
   - Los flujos de registro más recientes durante el tiempo de la migración
4. En estos flujos, busque palabras clave como:
   - «error» (error)
   - «exception» (excepción)
   - «failed» (fallo)
   - «warning» (advertencia)

Estado de la tarea de DMS (usando AWS CLI)

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

Flujo de estado previsto:

crear → listo → en ejecución → detener → detenido (o fallido)

Supervise mediante docdb-dashboarder

La `docdb-dashboarder` herramienta proporciona una supervisión completa de los clústeres de Amazon DocumentDB mediante la generación CloudWatch automática de paneles con métricas de rendimiento esenciales. Estos paneles muestran métricas críticas a nivel de clúster (retraso de réplica, contadores de operaciones), métricas a nivel de instancia (CPU, memoria, conexiones) y métricas de almacenamiento (uso del volumen, almacenamiento de copias de seguridad). Para las situaciones de migración, la herramienta ofrece paneles especializados que rastrean el progreso de la migración con métricas como el retraso en la replicación de la CDC y las tasas de operación. Los paneles pueden monitorear varios clústeres simultáneamente e incluir soporte para instancias. NVMe-backed Si visualiza estas métricas, los equipos pueden identificar de forma proactiva los cuellos de botella en el rendimiento, optimizar la asignación de recursos y garantizar el buen funcionamiento de sus implementaciones de Amazon DocumentDB. La herramienta elimina la necesidad de crear un panel manual y, al mismo tiempo, proporciona una supervisión uniforme en todos los entornos. Para obtener instrucciones de configuración y opciones de configuración avanzadas, consulte el repositorio de herramientas de panel de Amazon DocumentDB. GitHub

Validación

Temas

- Lista de comprobación de validación
- Validación de esquemas e índices
- Muestreo de datos y validación a nivel de campo
- Validación mediante herramienta DataDiffer

Esta sección proporciona un proceso de validación detallado para garantizar la coherencia, la integridad y la compatibilidad de las aplicaciones de los datos después de la migración.

Los pasos de validación se aplican independientemente del método de migración (AWS DMS u otras herramientas). mongodump/mongorestore

Lista de comprobación de validación

Compruebe que el número de documentos de cada colección coincide con el de origen y el de destino:

MongoDB como origen

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

Amazon DocumentDB como destino

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

Validación de esquemas e índices

Asegúrese de que:

- Todas las colecciones existen en el destino.
- Los índices se replican correctamente.
- Las definiciones de esquema (si se imponen) son idénticas.

Comprobación de colecciones (origen frente a destino)

```
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();"
```

Compruebe los índices (fuente frente a destino)

```
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()"
```

Compare la lista de colecciones para asegurarse de que no haya colecciones faltantes ni adicionales.

Verifique los índices. Para ello, compruebe los nombres de los índices, las definiciones clave, las restricciones únicas y los índices TTL (si los hubiera).

Comprobación de reglas de validación de esquemas (si usa la validación de esquemas en 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"
```

Muestreo de datos y validación a nivel de campo

Puede muestrear documentos de forma aleatoria y comparar campos entre el origen y el destino.

Muestreo manual

Obtenga cinco documentos aleatorios (origen):

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

Obtenga los mismos identificadores de documentos (destino):

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

Muestreo automático

```
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']}")
```

Validación mediante herramienta DataDiffer

La DataDiffer herramienta en sí GitHub proporciona una forma fiable de comparar los datos entre las bases de datos de origen y de destino.

Requisitos previos

Se deben cumplir los siguientes requisitos previos antes de instalar la DataDiffer herramienta:

- PythonMás de 3,7
- PyMongo biblioteca
- Conectividad de red con los clústeres de MongoDB de origen y de Amazon DocumentDB de destino

Instalación y configuración

Clona el repositorio y navega hasta el DataDiffer directorio

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

Instale las dependencias requeridas.

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

Ejecución de validación de datos

Cree un archivo de configuración (por ejemplo, config.json) con los detalles de la conexión.

```
{
  "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
  }
}
```

Ejecute la DataDiffer herramienta

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

En el caso de colecciones grandes, utilice el muestreo para validar un subconjunto de datos.

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

Para validar varias colecciones, cree archivos de configuración independientes o utilice el modo por lotes

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

Interpretación de los resultados

El resultado que mostrará la herramienta incluirá lo siguiente:

- Documentos totales de origen y destino
- Número de documentos coincidentes
- Número de documentos faltantes
- Número de documentos con diferencias
- Informe detallado de las diferencias (si las hubiera)

Prácticas recomendadas

Las siguientes son las prácticas recomendadas para usar la DataDiffer herramienta:

- Ejecute por fases: primero valide los recuentos de documentos, luego tome muestras de los documentos clave y, finalmente, ejecute una comparación completa, si es necesario.
- Compruebe si hay diferencias en los esquemas: Amazon DocumentDB tiene algunas limitaciones en comparación con MongoDB. La herramienta destacará los tipos o las estructuras de datos incompatibles.
- Valide durante los períodos de inactividad: ejecute la validación cuando las operaciones de escritura sean mínimas para garantizar la coherencia.
- Supervise el uso de los recursos: el proceso de comparación puede consumir muchos recursos. Ajuste el tamaño del lote y el número de subprocesos en consecuencia.
- Valide los índices: tras la validación de los datos, asegúrese de que se hayan creado todos los índices necesarios en el clúster de Amazon DocumentDB de destino.
- Resultados de la validación de documentos: mantenga un registro de los resultados de la validación de cada colección como parte de la documentación de migración.

Migración desde Couchbase Server

Temas

- Introducción
- Comparación con Amazon DocumentDB
- Discovery
- Planificación
- Migración
- Validación

Introducción

Esta guía presenta los puntos clave que hay que tener en cuenta al migrar de Couchbase Server a Amazon DocumentDB. En ella se explican las consideraciones que deben tenerse en cuenta para las fases de descubrimiento, planificación, ejecución y validación de la migración. También explica cómo realizar migraciones en línea y sin conexión.

Comparación con Amazon DocumentDB

	Servidor Couchbase	Amazon DocumentDB
Organización de datos	En las versiones 7.0 y posterior es, los datos se organizan en grupos, ámbitos y colecciones. En versiones anteriores, los datos se organizaban en grupos.	Los datos se organizan en bases de datos y colecciones.
Compatibilidad	Hay API independientes para cada servicio (por ejemplo, datos, índices, búsquedas, etc.). Las búsquedas secundarias utilizan SQL++ (antes conocido como N1QL), un lenguaje de consulta basado en ANSI-standard SQL, por lo que muchos desarrolladores lo conocen.	Amazon DocumentDB es compatible con la API de MongoDB. compatibility.html
Arquitectura	El almacenamiento se adjunta a cada instancia del clúster. No puede escalar el procesami ento de forma independiente del almacenamiento.	Amazon DocumentDB está diseñado para la nube y para evitar las limitaciones de las arquitecturas de bases de datos tradicionales. Las capas de procesamiento y almacenam iento están separadas en Amazon DocumentDB y la capa de procesamiento se puede escalar independientemente del almacenamiento.
Añada capacidad de lectura a petición	Los clústeres se pueden escalar añadiendo instancias. Como el almacenamiento se adjunta a la instancia en la que se ejecuta el servicio, el tiempo que se tarda en	Puede escalar la lectura de su clúster de Amazon DocumentD B creando hasta 15 réplicas de Amazon DocumentDB en el

	Servidor Couchbase	Amazon DocumentDB
	escalar depende de la cantidad de datos que sea necesario mover a la nueva instancia o reequilibrar.	clúster. No hay ningún impacto en la capa de almacenamiento.
Recupérese rápidamente de una falla en un nodo	Los clústeres tienen funciones de conmutación por error automática, pero el tiempo necesario para que el clúster recupere su capacidad máxima depende de la cantidad de datos que se deban mover a la nueva instancia.	Amazon DocumentDB puede realizar la conmutación por error de la instancia principal, normalmente en un plazo de 30 segundos, y restaurar el clúster a su máxima potencia en un plazo de 8 a 10 minutos, independi entemente de la cantidad de datos del clúster.
Amplíe el almacenam iento a medida que aumenten los datos	En el caso de los clústeres autogestionados, el almacenam iento y el iOS no se escalan automáticamente.	El almacenamiento de Amazon DocumentDB y iOS se escalan automáticamente.
Realice copias de seguridad de los datos sin afectar al rendimiento	Las copias de seguridad las realiza el servicio de copias de seguridad y no están habilitadas de forma predeterminada. Dado que el almacenamiento y el procesami ento no están separados, el rendimiento puede verse afectado.	Las copias de seguridad de Amazon DocumentDB están habilitadas de forma predeterm inada y no se pueden desactiva r. Las copias de seguridad las gestiona la capa de almacenam iento, por lo que no tienen ningún impacto en la capa de procesami ento. Amazon DocumentDB admite la restauración a partir de una instantánea de clúster y la restauración a un punto determina do.

	Servidor Couchbase	Amazon DocumentDB
Durabilidad de los datos	Puede haber un máximo de 3 réplicas de datos en un clúster para un total de 4 copias. Cada instancia en la que se ejecute el servicio de datos tendrá 1, 2 o 3 copias de réplica de los datos activas.	Amazon DocumentDB conserva 6 copias de los datos con independencia del número de instancias informáticas que haya, con un quórum de escritura de 4 y se mantiene fiel. Los clientes reciben un acuse de recibo después de que la capa de almacenamiento haya conservado 4 copias de los datos.
Coherencia	Se admite la coherencia inmediata de K/V las operaciones. El SDK de Couchbase dirige las K/V solicitud es a la instancia específica que contiene la copia activa de los datos, de modo que, una vez que se confirma una actualización, el cliente tiene la garantía de leerla. La replicación de las actualiza ciones en otros servicios (índice, búsqueda, análisis, generación de eventos) es, en última instancia, coherente.	En última instancia, las réplicas de Amazon DocumentDB son coherentes. Si se requieren lecturas de coherencia inmediata s, el cliente puede leer desde la instancia principal.
Replicación	Cross-Data La replicación central (XDCR) proporciona una replicaci ón filtrada, activa y passive/a ctive activa de datos en muchas: muchas topologías.	Los clústeres globales de Amazon DocumentDB proporcionan una replicación activa-pasiva en topologías 1:muchos (hasta 10).

Discovery

La migración a Amazon DocumentDB requiere un conocimiento profundo de la carga de trabajo de la base de datos existente. El descubrimiento de la carga de trabajo es el proceso de analizar la configuración y las características operativas del clúster de Couchbase (el conjunto de datos, los índices y la carga de trabajo) para garantizar una transición fluida con una interrupción mínima.

Configuración del clúster

Couchbase utiliza una arquitectura centrada en los servicios en la que cada capacidad corresponde a un servicio. Ejecute el siguiente comando en su clúster de Couchbase para determinar qué servicios se están utilizando (consulte Obtener información sobre los nodos):

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

Código de salida de ejemplo:

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

Los servicios de Couchbase incluyen lo siguiente:

Servicio de datos (kv)

El servicio de datos proporciona read/write acceso a los datos de la memoria y del disco.

Amazon DocumentDB admite K/V operaciones con datos JSON a través de la API de MongoDB.

Servicio de consultas (n1ql)

El servicio de consultas admite la consulta de datos de JSON a través de SQL++.

Amazon DocumentDB admite la consulta de datos de JSON mediante la API de MongoDB.

Servicio de índices (index)

El servicio de índices crea y mantiene índices de datos, lo que permite realizar consultas más rápidas.

Amazon DocumentDB admite un índice principal predeterminado y la creación de índices secundarios a partir de datos JSON mediante la API de MongoDB.

Servicio de búsqueda (fts)

El servicio de búsqueda admite la creación de índices para la búsqueda de texto completo.

La función nativa de búsqueda de texto completo de Amazon DocumentDB le permite realizar consultas de texto básicas en grandes conjuntos de datos textuales mediante índices de texto especiales mediante la API. MongoDB Para obtener funciones avanzadas como la búsqueda difusa, la búsqueda multilingüe y la búsqueda entre colecciones, utilice la integración Zero-ETL de Amazon DocumentDB con Amazon Service. OpenSearch

Servicio de análisis (cbas)

El servicio de análisis permite analizar los datos de JSON prácticamente en tiempo real.

Amazon DocumentDB admite consultas ad hoc sobre datos JSON mediante la API de MongoDB. También puede ejecutar consultas complejas en sus datos de JSON en Amazon DocumentDB mediante Apache Spark que se ejecuta en Amazon EMR.

Servicio de eventos (eventing)

El servicio de eventos ejecuta la lógica empresarial definida por el usuario en respuesta a los cambios en los datos.

Amazon DocumentDB automatiza las cargas de trabajo basadas en eventos al invocar AWS Lambda funciones cada vez que cambian los datos en su clúster de Amazon DocumentDB.

Servicio de respaldo (respaldo)

El servicio de respaldo programa copias de seguridad de datos completas e incrementales y combina las copias de seguridad de datos anteriores.

Amazon DocumentDB realiza copias de seguridad continuas de sus datos en Amazon S3 con un período de retención de 1 a 35 días para que pueda restaurarlos rápidamente en cualquier punto dentro del período de retención de la copia de seguridad. Amazon DocumentDB también toma instantáneas automáticas de los datos como parte de este proceso de copia de seguridad continua. También puede gestionar la copia de seguridad y la restauración de Amazon DocumentDB con. AWS Backup.

Características operativas

Utilice la herramienta de detección de Couchbase para obtener la siguiente información sobre su conjunto de datos, índices y carga de trabajo. Esta información le ayudará a dimensionar su clúster de Amazon DocumentDB.

Conjunto de datos

La herramienta recupera la siguiente información sobre el depósito, el alcance y la recopilación:

1. nombre del bucket
2. tipo de bucket
3. nombre del ámbito
4. nombre de la colección
5. tamaño total (bytes)
6. total de artículos
7. tamaño del artículo (bytes)

Índices

La herramienta recupera las siguientes estadísticas de índice y todas las definiciones de índice para todos los grupos. Tenga en cuenta que los índices principales están excluidos, ya que Amazon DocumentDB crea automáticamente un índice principal para cada colección.

1. nombre del bucket
2. nombre del ámbito
3. nombre de la colección
4. nombre del índice
5. tamaño del índice (bytes)

Carga de trabajo

La herramienta recupera las métricas de consulta K/V de N1QL. K/V Los valores de las métricas se recopilan a nivel de bucket y las métricas de SQL++ se recopilan a nivel de clúster.

Las opciones de línea de comandos de la herramienta son las siguientes:

```
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>
```

A continuación se muestra un comando de ejemplo:

```
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 Los valores métricos se basarán en muestras cada 10 minutos durante la semana anterior (consulte el método HTTP y el URI). Los valores métricos de SQL++ se basarán en muestras cada 1 segundo durante los últimos 60 segundos (consulte Etiquetas https://docs.couchbase.com/server/current/rest-api/rest-statistics-single.html#general-labels generales). El resultado del comando estará en los siguientes archivos:

collection-stats.csv: información sobre el bucket, el alcance y la colección

```
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 — nombres y tamaños de índices

```
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: obtiene, establece y elimina las métricas de todos los buckets

```
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++ selecciona, elimina e inserta métricas para el clúster

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

indexes- <bucket-name>.txt: definiciones de índice de todos los índices del bucket. Tenga en cuenta que los índices principales están excluidos, ya que Amazon DocumentDB crea automáticamente un índice principal para cada colección.

```
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`)
...
```

Planificación

En la fase de planificación, determinará los requisitos del clúster de Amazon DocumentDB y asignará los depósitos, ámbitos y colecciones de Couchbase a las bases de datos y colecciones de Amazon DocumentDB.

Requisitos de clúster de Amazon DocumentDB

Utilice los datos recopilados en la fase de descubrimiento para dimensionar su clúster de Amazon DocumentDB. Consulte Dimensionamiento de instancias para obtener más información sobre el tamaño de su clúster de Amazon DocumentDB.

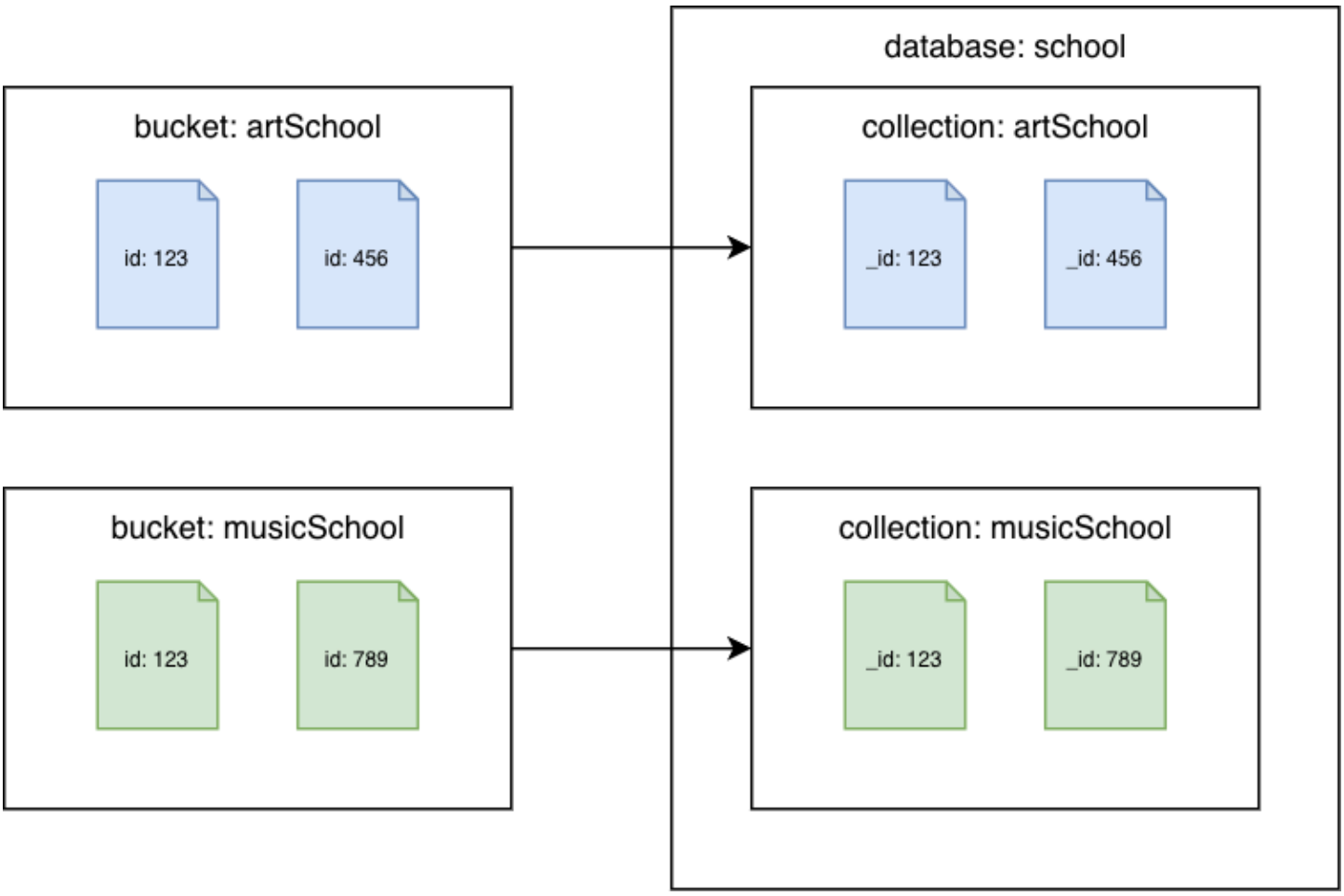
Asignación de cubos, ámbitos y colecciones a bases de datos y colecciones

Determine las bases de datos y colecciones que existirán en sus clústeres de Amazon DocumentDB. Tenga en cuenta las siguientes opciones en función de cómo estén organizados los datos en su clúster de Couchbase. Estas no son las únicas opciones, pero proporcionan puntos de partida que puede considerar.

Couchbase Server 6.x o anterior

Couchbase se dirige a las colecciones de Amazon DocumentDB

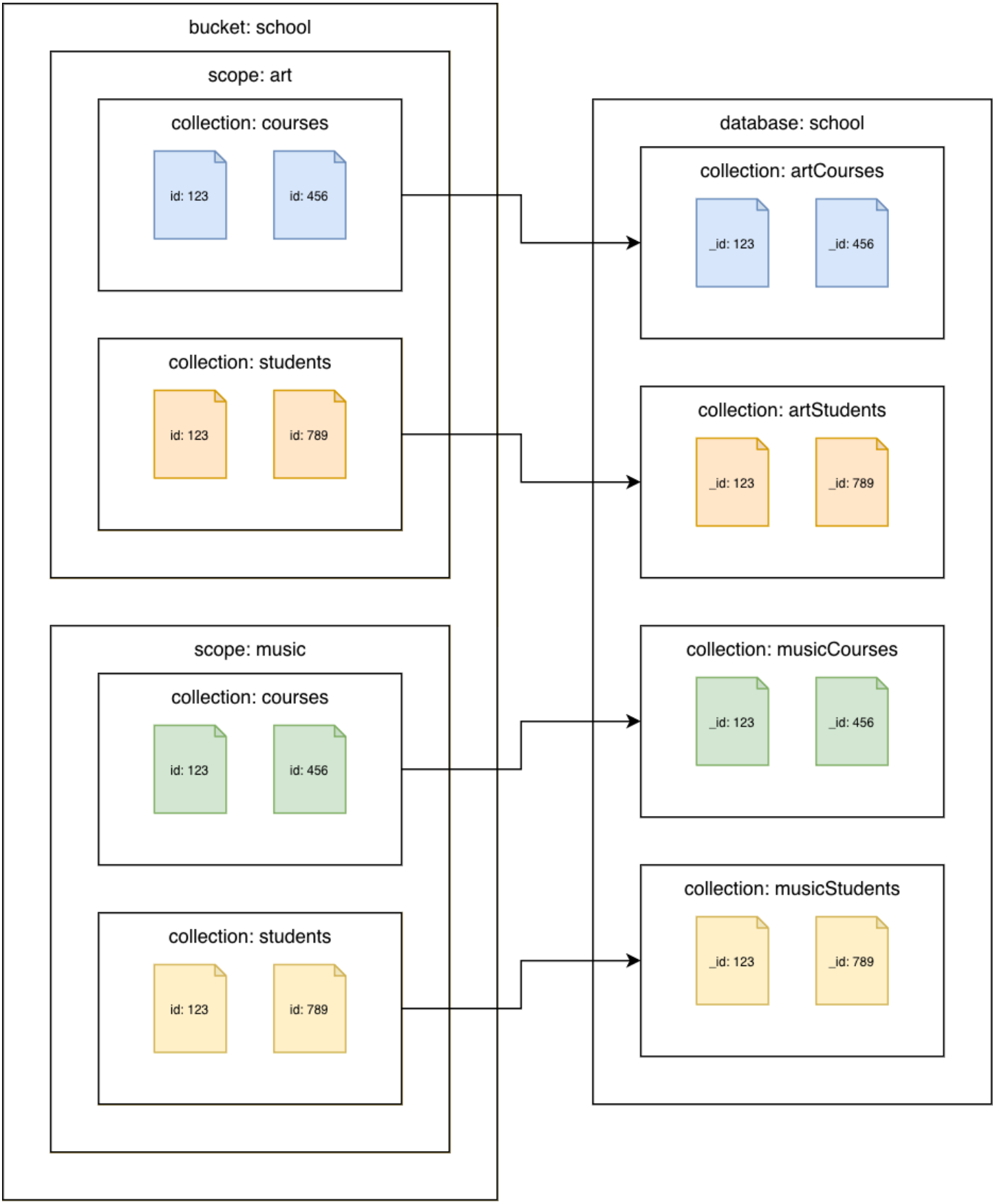
Migre cada bucket a una colección diferente de Amazon DocumentDB. En este escenario, el valor del documento de Couchbase se utilizará como el `id` valor de Amazon DocumentDB. `_id`



Couchbase Server 7.0 o posterior

De las colecciones de Couchbase a las colecciones de Amazon DocumentDB

Migre cada colección a una colección diferente de Amazon DocumentDB. En este escenario, el valor del documento de Couchbase se utilizará como el `id` valor de 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

Migración

Migración de índices

La migración a Amazon DocumentDB implica transferir no solo datos, sino también índices para mantener el rendimiento de las consultas y optimizar las operaciones de la base de datos. En esta sección se describe el proceso detallado paso a paso para migrar los índices a Amazon DocumentDB y, al mismo tiempo, garantizar la compatibilidad y la eficiencia.

Utilice Amazon Q (disponible en la consola de administración de AWS o en su IDE) para convertir las `CREATE INDEX` instrucciones de SQL++ en comandos de Amazon DocumentDB. `createIndex()`

1. Cargue los archivos <bucket name>indexes-.txt creados por la herramienta Discovery Tool para Couchbase.
2. Introduzca el siguiente mensaje:

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

Amazon Q generará `createIndex()` comandos de Amazon DocumentDB equivalentes. Tenga en cuenta que es posible que necesite actualizar los nombres de las colecciones en función de cómo asignó los depósitos, ámbitos y colecciones de Couchbase a las colecciones de Amazon DocumentDB.

Por ejemplo:

indexes-beer-sample.txt

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

Ejemplo de salida de Amazon Q (extracto):

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

```
)

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

Para ver los índices que Amazon Q no pueda convertir, consulte Administración de índices e índices de Amazon DocumentDB y de las propiedades de los índices para obtener más información.

Refactorice el código para usar las API de MongoDB

Los clientes usan los SDK de Couchbase para conectarse al servidor de Couchbase. Los clientes de Amazon DocumentDB utilizan los controladores MongoDB para conectarse a Amazon DocumentDB. Todos los idiomas compatibles con los SDK de Couchbase también son compatibles con los controladores de MongoDB. Consulte MongoDB Drivers para obtener más información sobre el controlador para su idioma.

Como las API de Couchbase Server y Amazon DocumentDB son diferentes, tendrá que refactorizar el código para usar las API de MongoDB adecuadas. Puede usar Amazon Q para convertir las llamadas a la API y las consultas de SQL++ en las K/V API de MongoDB equivalentes:

1. Cargue los archivos de código fuente.
2. Introduzca el siguiente mensaje:

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

Con el ejemplo de código de Python de Hello Couchbase, Amazon Q genera lo siguiente (extracto):

```
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")
```

Consulte la sección Conexión mediante programación a Amazon DocumentDB para ver ejemplos de cómo conectarse a Amazon DocumentDB en Python, PHP, Go, Java Node.js, C#/.NET, R y Ruby.

Seleccione el enfoque de migración

Al migrar datos a Amazon DocumentDB, hay dos opciones:

1. migración fuera de línea
2. migración en línea

> **Note**
>
> Amazon DocumentDB usa el `_id` campo como clave principal para cada documento. Los enfoques de migración de esta guía almacenan la clave del documento de Couchbase en el campo. `_id` Para la migración sin conexión, esto se realiza mediante la `--include-key _id` opción que aparece en. `cbexport` Para la migración en línea, esto se hace a través del conector Kafka. `ProvidedInKeyStrategy`

Migración sin conexión

Considere la posibilidad de realizar una migración offline cuando:

- El tiempo de inactividad es aceptable: la migración sin conexión implica detener las operaciones de escritura en la base de datos de origen, exportar los datos y, a continuación, importarlos a Amazon DocumentDB. Este proceso implica un tiempo de inactividad para su aplicación. Si su

aplicación o carga de trabajo pueden tolerar este período de falta de disponibilidad, la migración sin conexión es una opción viable.

- Migrar conjuntos de datos más pequeños o realizar pruebas de concepto: en el caso de los conjuntos de datos más pequeños, el tiempo necesario para el proceso de exportación e importación es relativamente corto, por lo que la migración sin conexión es un método rápido y sencillo. También es ideal para entornos de desarrollo, pruebas y pruebas de concepto en los que el tiempo de inactividad es menos crítico.
- La simplicidad es una prioridad: el método offline, que utiliza cbexport y mongoimport, suele ser el método más sencillo para migrar datos. Evita las complejidades de la captura de datos cambiados (CDC) que implican los métodos de migración en línea.
- No es necesario replicar los cambios en curso: si la base de datos de origen no recibe cambios de forma activa durante la migración, o si esos cambios no son fundamentales para capturarlos y aplicarlos al destino durante el proceso de migración, es apropiado adoptar un enfoque fuera de línea.

Temas

- Couchbase Server 6.x o anterior
- Couchbase Server 7.0 o posterior

Couchbase Server 6.x o anterior

Del paquete de Couchbase a la colección Amazon DocumentDB

Exporte los datos con cbexport json para crear un volcado en JSON de todos los datos del bucket. Para la `--format` opción puedes usar `lines` o. `list`

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

Importe los datos a una colección de Amazon DocumentDB mediante mongoimport con la opción adecuada para importar las líneas o la lista:

líneas:

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

lista:

```
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 posterior

Para realizar una migración sin conexión, utilice las herramientas cbexport y mongoimport:

Cubo de Couchbase con alcance y colección predeterminados

Exporte los datos con cbexport json para crear un volcado en JSON de todas las colecciones del bucket. Para la `--format` opción puedes usar `lines` o. `list`

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

Importe los datos a una colección de Amazon DocumentDB mediante mongoimport con la opción adecuada para importar las líneas o la lista:

líneas:

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

lista:

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

De las colecciones de Couchbase a las colecciones de Amazon DocumentDB

Exporte los datos con cbexport json para crear un volcado en JSON de cada colección. Usa la `--include-data` opción para exportar cada colección. Para la `--format` opción, puede usar `lines` `olist`. Usa las `--collection-field` opciones `--scope-field` y para almacenar el nombre del ámbito y la colección en los campos especificados de cada 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"
```

Como cbexport agregó los `_collection` campos `_scope` y a todos los documentos exportados, puedes eliminarlos de todos los documentos del archivo de exportación mediante la búsqueda y el reemplazo`sed`, o mediante el método que prefieras.

Importe los datos de cada colección a una colección de Amazon DocumentDB mediante mongoimport con la opción adecuada para importar las líneas o la lista:

líneas:

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

lista:

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

Migración en línea

Considere la posibilidad de realizar una migración en línea cuando necesite minimizar el tiempo de inactividad y los cambios en curso deban replicarse en Amazon DocumentDB prácticamente en tiempo real.

Consulte Cómo realizar una migración en vivo de Couchbase a Amazon DocumentDB para obtener información sobre cómo realizar una migración en vivo a Amazon DocumentDB. La documentación explica cómo implementar la solución y realizar una migración en vivo de un bucket a un clúster de Amazon DocumentDB.

Temas

- Couchbase Server 6.x o anterior
- Couchbase Server 7.0 o posterior

Couchbase Server 6.x o anterior

Del paquete de Couchbase a la colección Amazon DocumentDB

La utilidad de migración de Couchbase está preconfigurada para realizar una migración en línea de un bucket de Couchbase a una colección de Amazon DocumentDB. Si analizamos la configuración del conector de sumidero, el `document.id.strategy` parámetro está configurado para usar el valor de la clave del mensaje como valor del `_id` campo (consulte Propiedades de la estrategia del identificador del conector de sumidero):

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

Couchbase Server 7.0 o posterior

Cubeta de Couchbase con alcance y recopilación predeterminados

La utilidad de migración de Couchbase está preconfigurada para realizar una migración en línea de un bucket de Couchbase a una colección de Amazon DocumentDB. Si analizamos la configuración del conector de sumidero, el `document.id.strategy` parámetro está configurado para usar el valor de la clave del mensaje como valor del `_id` campo (consulte Propiedades de la estrategia del identificador del conector de sumidero):

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

De las colecciones de Couchbase a las colecciones de Amazon DocumentDB

Configure el conector de código fuente para transmitir cada colección de Couchbase de cada ámbito a un tema diferente (consulte las opciones de configuración del código fuente). Por ejemplo:

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

Configure el conector de sumidero para transmitir desde cada tema a una colección independiente de Amazon DocumentDB (consulte Propiedades de configuración del conector de sumidero). Por ejemplo:

```
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>'
```

Validación

Esta sección proporciona un proceso de validación detallado para verificar la coherencia e integridad de los datos después de la migración a Amazon DocumentDB. Los pasos de validación se aplican independientemente del método de migración.

Temas

- Verifique que todas las colecciones existan en el destino
- Verifique el recuento de documentos entre los clústeres de origen y destino
- Compare los documentos entre los clústeres de origen y de destino

Verifique que todas las colecciones existan en el destino

Fuente de Couchbase

opción 1: banco de trabajo de consultas

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

opción 2: herramienta cbq

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

Amazon DocumentDB como destino

mongosh (consulte Conectarse a su clúster de Amazon DocumentDB):

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

Verifique el recuento de documentos entre los clústeres de origen y destino

Fuente de Couchbase

Couchbase Server 6.x o anterior

opción 1: banco de trabajo de consultas

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

opción 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 posterior

opción 1: banco de trabajo de consultas

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

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

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

Amazon DocumentDB como destino

mongosh (consulte Conectarse a su clúster de Amazon DocumentDB):

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

Compare los documentos entre los clústeres de origen y de destino

Fuente de Couchbase

Couchbase Server 6.x o anterior

opción 1: banco de trabajo de consultas

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

opción 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 posterior

opción 1: banco de trabajo de consultas

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

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

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

Amazon DocumentDB como destino

mongosh (consulte Conectarse a su clúster de Amazon DocumentDB):

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

Fechas de soporte de la versión del motor de Amazon DocumentDB

Puede utilizar las siguientes fechas para planificar sus ciclos de prueba y actualización.

Calendario de lanzamientos de las principales versiones de Amazon DocumentDB

Amazon DocumentDB admite actualmente las siguientes versiones principales. Para actualizar la versión del motor, consulte Actualización de una versión principal del motor oActualización mediante AWS DMS.

Engine version (Versión del motor)	Fecha de publicación	Finalización del soporte estándar	Inicio del soporte extendido (precio del primer año)	Inicio del soporte extendido (precio del tercer año)	Fecha de finalización del soporte extendido
Versión 3.6	9 de enero de 2019	30 de marzo de 2026	31 de marzo de 2026	31 de marzo de 2028	30 de marzo de 2029
Versión 4.0	9 de noviembre de 2020	N/A	N/A	N/A	N/A
Versión 5.0	1 de marzo de 2023	N/A	N/A	N/A	N/A
Versión 8.0	14 de noviembre de 2025	N/A	N/A	N/A	N/A

Amazon DocumentDB admite varias versiones de motores en el marco del soporte estándar. Puede seguir ejecutando una versión después de la fecha de finalización del soporte estándar con el pago de una tarifa por Soporte extendido. Para obtener más información, consulte Soporte extendido de Amazon DocumentDB y Precios de Amazon DocumentDB.

Calendario de lanzamientos de las versiones secundarias de Amazon DocumentDB

Amazon DocumentDB admite actualmente las siguientes versiones secundarias. El calendario de lanzamiento variará en función de las funciones o correcciones adicionales. Las versiones

secundarias pueden llegar al final del soporte estándar antes de que lo hagan las versiones principales. Para actualizar la versión del motor, consulteActualización de una versión secundaria del motor.

> **Note**
>
> Las versiones secundarias están disponibles a partir de la versión principal 5.0.

Engine version (Versión del motor)	Fecha de publicación
Versión 5.0.0 (LTS)	16 de febrero de 2026
Versión 5.0.1	8 de junio de 2026

Amazon DocumentDB designa determinadas versiones como versiones Long-Term Support (LTS). Para obtener más información, consulte Uso de una versión de soporte a largo plazo (LTS).

Actualización local de la versión principal Amazon DocumentDB

Puede realizar una actualización local de la versión principal (MVU) de su clúster de Amazon DocumentDB manteniendo los mismos puntos de enlace, almacenamiento y etiquetas. Sus aplicaciones siguen funcionando sin modificaciones. Esta función está disponible sin coste adicional en todos los Regiones de AWS lugares en los que Amazon DocumentDB esté disponible.

> **Important**
>
> El clúster no está disponible durante la actualización y se reinicia varias veces. No se conecte al clúster, no lea desde él ni escriba en él después de iniciar la actualización. El tiempo de inactividad varía en función del número de recopilaciones, índices, bases de datos e instancias. Realice la actualización durante el período de mantenimiento o durante las horas de baja utilización.
>
> Una vez actualizada, no puede cambiar a una versión anterior. Si es necesario, puede restaurar la instantánea anterior a la actualización en un clúster nuevo.

Temas

- Rutas de actualización admitidas
- Requisitos previos
- Prácticas recomendadas
- Qué cambia tras la actualización a Amazon DocumentDB 8.0
- Post-upgrade consideraciones para los clústeres actualizados desde la versión 3.6 o 4.0
- Aplicación de la actualización
- Resolución de problemas

Rutas de actualización admitidas

En la siguiente tabla se enumeran las versiones del motor de destino que puede seleccionar para cada versión de origen. No es necesario actualizar primero a la versión `.0` secundaria y, a continuación, aplicar una actualización a la versión secundaria.

Versión de fuente	Versiones de destino compatibles
3.6	Cualquier versión secundaria 5.0 u 8.0 publicada
4.0	Cualquier versión secundaria 5.0 u 8.0 publicada
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	80.2

Para conocer las consideraciones posteriores a la actualización, consulte the section called “Post-upgrade consideraciones para los clústeres actualizados desde la versión 3.6 o 4.0” y. the section called “Qué cambia tras la actualización a Amazon DocumentDB 8.0”

Para ver las versiones secundarias disponibles en su AWS región, utilice el AWS CLI comando`aws docdb describe-db-engine-versions`. Para obtener una lista de las versiones secundarias publicadas, consulteNotas de la versión.

> **Note**
>
> In-place MVU no es compatible con clústeres globales o elásticos. Para actualizar un clúster global, elimine los clústeres secundarios, convierta el principal en un clúster regional, ejecute la MVU y, a continuación, vuelva a crear el clúster global añadiendo clústeres secundarios con los mismos nombres para conservar los puntos finales. Se le cobrarán I/O cargos mientras el servidor principal actualizado replica los datos en los nuevos servidores secundarios. Para ver los pasos detallados, consulte Eliminación de un clúster global de Amazon DocumentDB.

Requisitos previos

> **Important**
>
> Amplíe las instancias con ráfagas antes de realizar la actualización. Si tu clúster usa tipos de instancias escalables (por ejemplo, `db.t3.medium` o`db.t4g.medium`), amplía la instancia principal al menos `db.r5.large` o `db.r6g.large` antes de iniciar la actualización. Es posible que las instancias en ráfagas no tengan suficiente CPU y memoria para completar el proceso de actualización, lo que puede provocar errores en la actualización y una mayor falta de disponibilidad del clúster. Puede reducir la escala una vez finalizada la actualización.
> Amplíe la instancia de escritura sin servidor antes de la actualización. Si el clúster tiene una instancia de escritura sin servidor, la DCU máxima del clúster `ServerlessV2ScalingConfiguration` debe establecerse en 3 o más antes de iniciar la actualización. Los clústeres con una DCU máxima inferior a 3 no se pueden actualizar porque no tienen suficiente CPU o memoria para completar la actualización.
> Compruebe los índices parciales antes de realizar la actualización (actualice de Amazon DocumentDB 5.0 a 8.0). Antes de iniciar la actualización, revise los índices parciales para ver si hay uno `partialFilterExpression` que utilice el `$type` operador para coincidir con ciertos tipos de BSON: específicamente`binData`, los alias`bool`,`date`, `double`,`decimal`, y de `array` tipo, o un código de tipo numérico, como. `{ "$type" : 1 }` Estos índices parciales hacen que la actualización falle, aunque el índice funcione normalmente para lectura y escritura. `db.collection.getIndexes()`Utilícelo para revisar `partialFilterExpression` los índices parciales. Elimine el índice parcial afectado antes de iniciar la actualización y vuelva a crearlo cuando finalice la actualización.

Si no está seguro de si un índice está afectado, póngase en contacto con el servicio de AWS asistencia antes de realizar la actualización.

- Tipo de instancia: Amazon DocumentDB 4.0+ no admite instancias db.r4. Modifique las instancias para `db.r4.*` convertirlas en instancias o versiones más recientes antes de `db.r5.*` actualizarlas. Consulte Modificación de una instancia de base de datos de Amazon DocumentDB y Clases de instancias admitidas por región.
- Parches del sistema operativo: aplica cualquier acción de mantenimiento del sistema operativo pendiente en todas las instancias antes de la actualización. Consulte Actualizaciones del sistema operativo de Amazon DocumentDB.

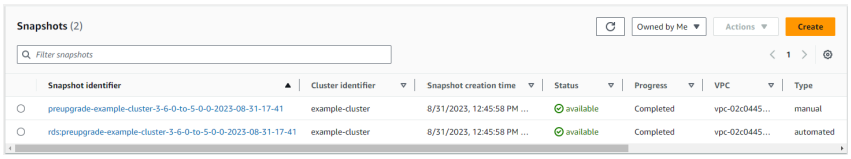
> **Note**
>
> Los parches pendientes del motor a nivel de clúster pueden ocultar los parches del sistema operativo de la instancia. Aplica primero los parches del motor si es necesario. Consulte Actualización de un parche a la versión del motor de un clúster.

- Límites de indexación para las instancias en ráfagas (instancias de la familia T): si tienes más de 3000 índices en instancias en ráfagas, amplía la instancia principal a al menos db.r5.large antes de actualizarla. Puedes reducir la escala una vez finalizada la actualización.

Instancia	Índices máximos para MVU
db.t4g.medium	3,000
db.t3.medium	10,000

- Grupo de parámetros: tenga listo un grupo de parámetros de clúster personalizado para la versión de destino antes de la actualización. Si no se especifica ninguno, se utilizará el grupo de parámetros predeterminado para la versión de destino (por ejemplo, `default.docdb5.0` o`default.docdb8.0`).
- Instantánea manual: cree una instantánea manual antes de la actualización. El proceso de actualización crea una instantánea automática denominada`preupgrade-<name>-<version>-<timestamp>`, pero siempre crea tu propia copia de seguridad. Consulte Creación de una instantánea manual del clúster.

> **Note**
>
> La instantánea automática creada por el proceso de actualización no se eliminará automáticamente una vez que se haya completado la actualización local de la versión principal. Esta instantánea no incurrirá en ningún cargo mientras se encuentre dentro del período de retención. Puede optar por eliminar esta instantánea una vez que haya comprobado que la actualización del clúster se ha realizado correctamente.



Prácticas recomendadas

Antes de la actualización: prueba con un clon

1. Clonación de un volumen de clúster de base de datos de Amazon DocumentDBUtilízalo para crear un clon de tu clúster. No incurrirás en costos de almacenamiento a menos que modifiques los datos del clon.
2. Haga coincidir el recuento de instancias del clon con el clúster de destino para obtener una estimación temporal realista.
3. Realice la MVU en el clon y compruebe completamente las diferencias funcionales.
4. Para comprobar si ya hay una actualización programada, ejecute `describe-db-clusters` y busque`PendingModifiedValues.EngineVersion`. Si modificó el clúster y decidió aplicarlo en la siguiente ventana de mantenimiento, la programación no aparecerá en la consola, pero podrá verla en AWS CLI:

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

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

5. Si la prueba se realiza correctamente, continúe con la actualización del clúster de producción.

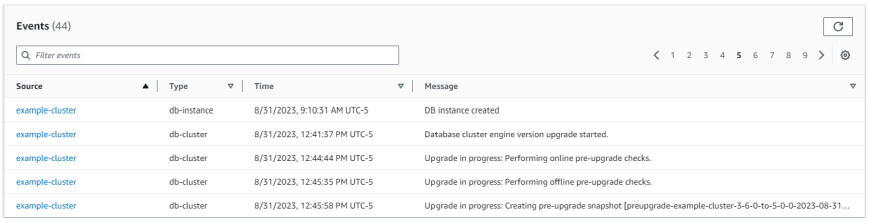
Durante la actualización

Puede supervisar el progreso de la actualización local de la versión principal suscribiéndose a los eventos de mantenimiento del clúster. Cuando finalice la actualización, recibirá el mensaje «Se ha actualizado la versión principal del clúster de bases de datos». Este y otros eventos que se producen durante la actualización aparecen en la sección Eventos y etiquetas de la página de detalles del clúster de la consola de Amazon DocumentDB. A continuación, el estado del clúster cambia de `upgrading` `aavailable`.

Durante la actualización se generan los siguientes eventos:

1. Se inició la actualización de la versión principal del motor de clúster de bases de datos. El clúster permanece en línea.
2. Preparación de la actualización en curso: se están iniciando las comprobaciones previas de la actualización en línea.
3. Preparación de la actualización en curso: se han completado las comprobaciones previas de la actualización en línea.
4. Desconectar el clúster de bases de datos mientras la instancia principal completa el patch/upgrade proceso.
5. La preparación de la actualización está en curso: al iniciar la actualización sin conexión se realizan comprobaciones previas.
6. Preparación de la actualización en curso: se han completado las comprobaciones previas a la actualización sin conexión.
7. **<version-to><timestamp>**Actualización en curso: creación de una instantánea previa a la actualización [preupgrade- **<cluster-name>**- **<version-from>**-to- -].
8. Actualización en curso: volumen de clonación.
9. Actualización en curso: actualización del escritor.
10. Actualización en curso: actualización de los lectores.
11. Se ha actualizado la versión principal del motor de clústeres de bases de datos.

Los eventos están visibles en la consola, en la página de eventos:



Desde el AWS CLI, puede ejecutar `aws docdb describe-events` para supervisar el progreso de la actualización. Para recibir notificaciones automáticamente, utilice la opción `aws docdb create-event-subscription` para suscribirse a los eventos y enviarlos a Amazon SNS para recibir correos electrónicos, mensajes push u otros métodos de entrega. Para obtener más información, consulte Suscripción a eventos de Amazon DocumentDB.

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

El comando devuelve un resultado similar al siguiente:

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

Después de la actualización

> **Warning**
>
> Inmediatamente después de la actualización local de la versión principal, su clúster de Amazon DocumentDB rellena los metadatos del índice que el motor de base de datos utiliza para optimizar los planes de ejecución de consultas. El rendimiento de las consultas vuelve

a los niveles esperados una vez finalizado este proceso. Por lo general, finaliza en unos minutos, pero puede tardar hasta dos horas, según la cantidad de índices del clúster.
No reinicie, conmute por error ni escale la instancia de up/down escritura durante este tiempo, ya que podría interrumpir el recálculo de los metadatos del índice. Espere hasta que observe el rendimiento esperado de la consulta antes de realizar dichos cambios.
Realice un seguimiento del progreso mediante los siguientes eventos de clúster:

- Post-upgrade estado del clúster: se inició el proceso de actualización de los metadatos del índice
- Post-upgrade estado del clúster: el proceso de actualización de los metadatos del índice se completó en X segundos

Ponte en contacto con el servicio de AWS asistencia si el proceso de actualización de los metadatos del índice no se ha completado en tres horas o si sigues teniendo problemas de rendimiento una vez finalizado el proceso.

1. Realiza una instantánea manual del clúster actualizado en caso de que necesites restaurarlo al estado posterior a la actualización. El proceso de creación automática de instantáneas se reanudará tan pronto como finalice la actualización local de la versión principal.
2. Clústeres de etiquetas actualizados desde la versión 3.6. Agregue una etiqueta para diferenciar los clústeres actualizados desde la versión 3.6. (consultethe section called “Post-upgrade consideraciones para los clústeres actualizados desde la versión 3.6 o 4.0”).
3. Actualiza tu controlador. Para usar nuevas funciones (por ejemplo, intercalación, vistas o compresión Zstd en 8.0), actualice a la versión correspondiente de la API de MongoDB. Para obtener más información, consulte Novedades de Amazon DocumentDB 8.0.
4. Pruébelo minuciosamente. Valide su aplicación con el clúster actualizado.

Qué cambia tras la actualización a Amazon DocumentDB 8.0

Tras actualizar una versión principal a Amazon DocumentDB 8.0 (desde Amazon DocumentDB 5.0 o directamente desde la 3.6 o 4.0), se habilitan o modifican las siguientes funciones:

- Intercalación. Amazon DocumentDB 8.0 admite la ??? intercalación. Tras la actualización, las colecciones nuevas y sus índices, así como los nuevos índices de las colecciones existentes, tienen habilitada la intercalación de forma predeterminada.

- Índice de texto. Los nuevos índices de texto se crean con Text Index V2, que utiliza un analizador de búsqueda de texto actualizado para mejorar la compatibilidad con MongoDB. Los índices de texto existentes no se ven afectados.
- Versión del planificador de consultas. Si no tenía un grupo de parámetros personalizado, se crea un nuevo grupo de parámetros predeterminado para Amazon DocumentDB 8.0 con la versión 3 de Planner seleccionada automáticamente. Con la versión 3 del planificador de consultas, las vistas también están disponibles.
- Compresión. Amazon DocumentDB 8.0 admite la compresión de documentos basada en diccionarios mediante el algoritmo Zstd. Tras la actualización, se crean nuevas colecciones con la compresión Zstd habilitada de forma predeterminada. Las colecciones existentes de la versión 5.0 conservan su configuración de compresión. Para aprovechar la compresión Zstd en las colecciones existentes, puede modificar su configuración de compresión. Para obtener más información, consulte Administración de la compresión basada en diccionarios en Amazon DocumentDB 8.0.
- Reconstrucción del índice. Si va a actualizar de Amazon DocumentDB 5.0 a Amazon DocumentDB 8.0, no es necesario reconstruir el índice. Si va a actualizar directamente de Amazon DocumentDB 3.6 o 4.0 a 8.0, reconstruya los índices tal y como se describe en. the section called "Post-upgrade consideraciones para los clústeres actualizados desde la versión 3.6 o 4.0"

> **Important**
>
> Amazon DocumentDB 8.0 requiere TLS 1.2 o superior. Ya no se admiten TLS 1.0 y TLS 1.1.

> **Note**
>
> Para obtener una lista completa de las diferencias funcionales, consulte. Compatibilidad de Amazon DocumentDB con MongoDB

Post-upgrade consideraciones para los clústeres actualizados desde la versión 3.6 o 4.0

- Reconstrucción del índice. Una MVU conserva los índices originales. Amazon DocumentDB 5.0 y versiones posteriores han mejorado el mantenimiento de los índices y la recolección de basura,

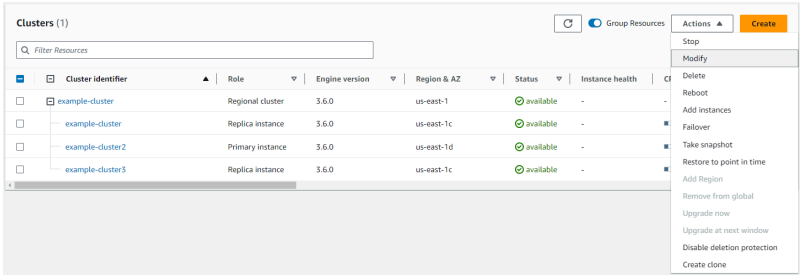
especialmente para los índices de baja cardinalidad. Tras actualizar la versión 3.6 o 4.0 (a la 5.0 o la 8.0), reconstruya los índices para garantizar un rendimiento óptimo de las consultas (opcional, implica un adicional). I/O Consulte Mantenimiento de índices mediante ReIndex .

- Comparación numérica de subdocumentos (solo 3.6). Los clústeres actualizados desde la versión 3.6 heredan el comportamiento de la versión 3.6, según el cual los tipos numéricos de los subdocumentos no se comparan entre tipos. Por ejemplo, `{a: {b: NumberLong(1)}}` no es igual `{a: {b: 1}}` en 3.6, pero es igual en 4.0 y versiones posteriores. Este comportamiento afecta a todos los clústeres actualizados desde la versión 3.6.

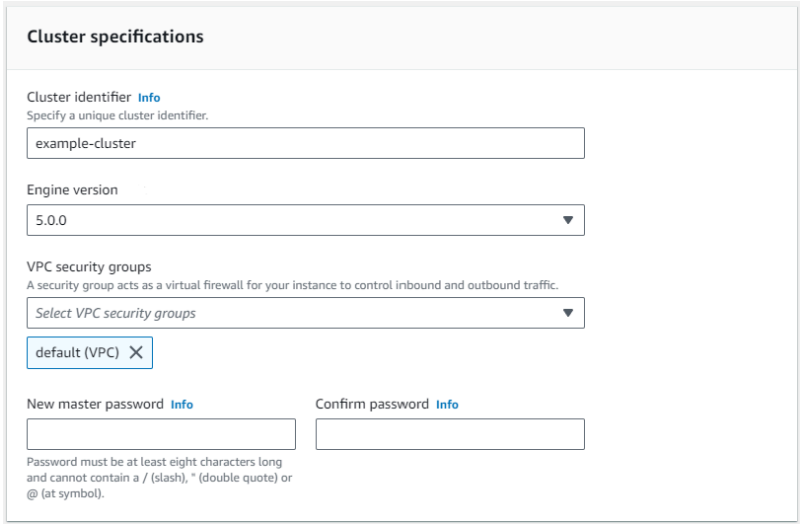
Aplicación de la actualización

Using the Consola de administración de AWS

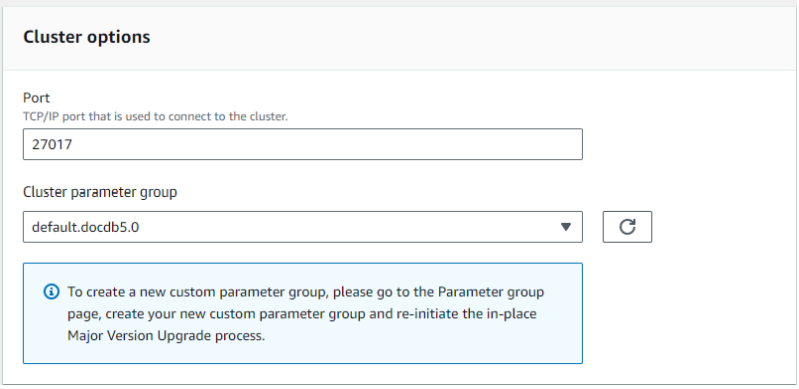
1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.
2. En la tabla Clústeres, seleccione el clúster de origen, elija Acciones y, a continuación, Modificar.



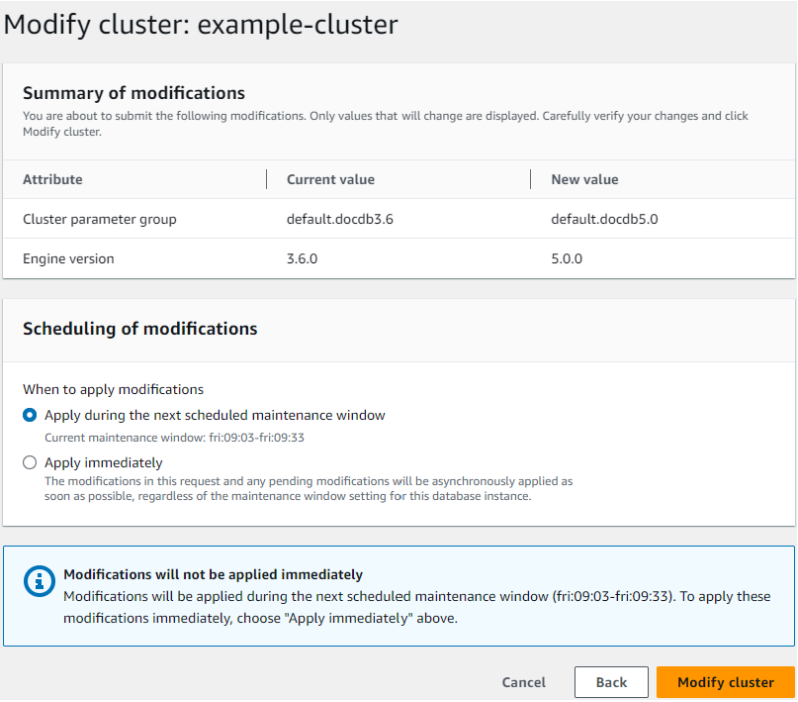
3. En las especificaciones del clúster, elija la versión de destino (por ejemplo, 5.0.0 u 8.0.0) en el menú desplegable de versiones del motor.



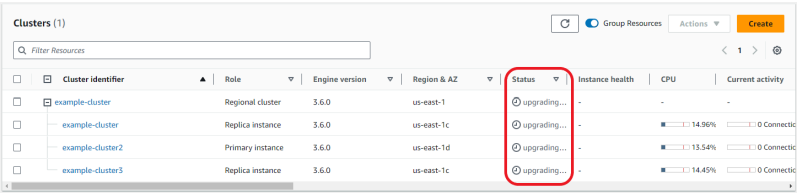
4. En las opciones del clúster, selecciona el grupo de parámetros del clúster para la versión del motor de destino. Puedes usar el grupo de parámetros predeterminado (por ejemplo, default.docdb5.0 o default.docdb8.0) o un grupo de parámetros personalizado que hayas creado.



5. Elija Continuar, seleccione su preferencia de programación (aplicar inmediatamente o en el siguiente período de mantenimiento) y, a continuación, elija Modificar clúster.



6. Supervise el estado del clúster en la tabla de clústeres a medida que pasa a actualizarse:



Using the AWS CLI

`modify-db-cluster`Utilízalo con la `--allow-major-version-upgrade` bandera:

```
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
```

Sustituya cada *placeholder* una por la información de su clúster.

Resolución de problemas

- Pre-upgrade error de comprobación. Antes de que comience la actualización, Amazon DocumentDB ejecuta comprobaciones de validación previas a la actualización. Las siguientes son las causas más comunes de errores en las comprobaciones previas:
  - La actualización no pudo continuar porque los nombres de las colecciones tienen 58 caracteres o más. Cambie el nombre de las colecciones afectadas por nombres más cortos antes de volver a intentar la actualización.
  - La actualización no pudo continuar porque el recuento de índices supera el límite establecido para el tipo de instancia. Actualice a un tipo de instancia más grande antes de volver a intentar la actualización. Para ver los límites de índice por tipo de instancia, consulteRequisitos previos.
- Fallo de actualización y reversión. Si se produce un error en la actualización, intentará revertirla automáticamente. Si la reversión se realiza correctamente, se produce el siguiente evento: «El clúster de la base de datos se encuentra en un estado que no se puede actualizar». El clúster vuelve a su estado anterior a la actualización y puede seguir utilizándolo. Ponte en contacto con AWS el servicio de asistencia para solucionar el problema antes de volver a intentarlo.
- Post-upgrade rendimiento. Es posible que se produzca una degradación temporal del rendimiento y un uso elevado de la CPU mientras se ejecuta la actualización de los metadatos del índice. Si la degradación persiste durante más de 3 horas, póngase en contacto con el servicio de AWS asistencia.

Para obtener más ayuda, póngase en contacto con. AWS Support

Actualización de la versión secundaria de Amazon DocumentDB

> **Note**
>
> En este tema se tratan las actualizaciones de versiones secundarias dentro de la misma versión principal (por ejemplo, la actualización de la 5.0.0 a la 5.0.1). Para obtener información sobre la actualización a una versión principal diferente, consulte. Actualización local de la versión principal Amazon DocumentDB

Puede actualizar la versión secundaria del motor de su clúster de Amazon DocumentDB (por ejemplo, de 5.0.0 a 5.0.1) modificando el clúster. Las actualizaciones de versiones menores se pueden aplicar inmediatamente o durante el siguiente período de mantenimiento.

Antes de realizar una actualización de versión secundaria

Antes de realizar una actualización de una versión menor, las siguientes acciones pueden resultar útiles:

- Realice la actualización durante un período de poco tráfico para reducir el impacto de cualquier tiempo de inactividad breve.
- Compruebe que la versión secundaria de destino sea un objetivo de actualización válido para su versión actual mediante el `describe-db-engine-versions` comando.
- En el caso de los clústeres globales, los clústeres secundarios deben actualizarse antes que el clúster principal. Para obtener más información, consulte Aplicación de parches a clústeres globales.

> **Important**
>
> - Las actualizaciones son operaciones unidireccionales: no se puede bajar de categoría después de la actualización.
> - El clúster experimentará un breve tiempo de inactividad durante el proceso de actualización.

Actualización a una versión secundaria

Using the Consola de administración de AWS

Para realizar una actualización de una versión menor mediante Consola de administración de AWS:

1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.
2. En la tabla de clústeres, seleccione el clúster que desee actualizar, elija Acciones y, a continuación, seleccione Modificar.
3. En el cuadro de diálogo Modificar el clúster, en la sección Especificaciones del clúster, seleccione la versión secundaria de destino en el menú desplegable Versión del motor.
4. Desplácese hacia abajo y seleccione Continuar.
5. En la sección Programación de modificaciones, elija aplicar el cambio inmediatamente o durante el siguiente período de mantenimiento.

   A continuación, seleccione Modify cluster (Modificar clúster).

Using the AWS CLI

Utilice el comando `modify-db-cluster` con el parámetro `--engine-version`:

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

En el ejemplo anterior, sustituya cada uno *`user input placeholder`* por la información de su clúster.

> **Note**
>
> Para aplicar la actualización durante el siguiente período de mantenimiento en lugar de hacerlo inmediatamente, `--no-apply-immediately` utilícela en lugar de`--apply-immediately`.

Verificar la actualización de la versión secundaria

Una vez completada la actualización, compruebe la versión del motor mediante: AWS CLI

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

La salida de esta operación será similar a lo que se indica a continuación:

```
"5.0.1"
```

Actualización del clúster de Amazon DocumentDB mediante AWS Database Migration Service

> **Important**
>
> Amazon DocumentDB no sigue los mismos ciclos de vida de soporte que MongoDB y el calendario de fin de ciclo de vida de MongoDB no es aplicable a Amazon DocumentDB.

Puede actualizar su clúster de Amazon DocumentDB a una versión superior con un tiempo de inactividad mínimo utilizando. AWS DMS AWS DMS es un servicio totalmente gestionado que facilita la migración desde versiones anteriores de Amazon DocumentDB, bases de datos relacionales y bases de datos no relacionales al clúster de Amazon DocumentDB de destino.

Temas

- Paso 1: habilitación de los flujos de cambios
- Paso 2: modificación de la duración de la retención de los flujos de cambios
- Paso 3: migración de los índices
- Paso 4: Cree un AWS DMS instancia de replicación
- Paso 5: Cree un AWS DMS punto final de origen
- Paso 6: Crear una AWS DMS punto final de destino
- Paso 7: crear y ejecutar una tarea de migración

- Paso 8: cambiar el punto de conexión de origen de la aplicación al clúster de Amazon DocumentDB de destino

Paso 1: habilitación de los flujos de cambios

Para realizar una migración con un tiempo de inactividad mínimo, AWS DMS requiere acceso a los flujos de cambios del clúster. La secuencia de cambios de Amazon DocumentDB brinda una secuencia en orden cronológico de los eventos de actualización que se producen dentro de las colecciones y bases de datos de su clúster. La lectura del flujo de cambios permite AWS DMS realizar una captura de datos de cambios (CDC) y aplicar actualizaciones incrementales al clúster de Amazon DocumentDB de destino.

Para habilitar las secuencias de cambios para todas las colecciones de una base de datos específica, autentíquese en su clúster de Amazon DocumentDB mediante el intérprete de comandos de mongo y ejecute los siguientes comandos:

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

Paso 2: modificación de la duración de la retención de los flujos de cambios

A continuación, modifique el período de retención de la secuencia de cambios en función del tiempo que desee conservar los eventos de cambio en la secuencia de cambios. Por ejemplo, si espera que la migración del clúster de Amazon DocumentDB AWS DMS tarde 12 horas, debe establecer la retención del flujo de cambios en un valor superior a 12 horas. El periodo de retención predeterminado para el clúster de Amazon DocumentDB es de tres horas. Puede modificar la duración de retención del registro de flujos de cambios de su clúster de Amazon DocumentDB para que oscile entre una hora y siete días utilizando el Consola de administración de AWS o el. AWS CLI Para obtener más información, consulte Modificación de la duración de la retención del registro de flujos de cambios.

Paso 3: migración de los índices

Cree los mismos índices en el clúster de Amazon DocumentDB de destino que en el clúster de Amazon DocumentDB de origen. Si bien AWS DMS gestiona la migración de datos, no migra los índices. Para migrar los índices, utilice la herramienta de índices de Amazon DocumentDB para

exportar los índices del clúster de Amazon DocumentDB de origen. Para obtener la herramienta, cree un clon del GitHub repositorio de herramientas de Amazon DocumentDB y siga las instrucciones que se indican en. `README.md` Puede ejecutar la herramienta desde una instancia de Amazon EC2 o desde un AWS Cloud9 entorno que se ejecute en la misma Amazon VPC que su clúster de Amazon DocumentDB.

Para utilizar el ejemplo siguiente, sustituya *`user input placeholder`* con su propia información.

El siguiente código descarga los índices del clúster de Amazon DocumentDB de origen:

```
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 vez que los índices se hayan exportado correctamente, restaure esos índices en el clúster de Amazon DocumentDB de destino. Para restaurar los índices que exportó en el paso anterior, utilice la herramienta de índices de Amazon DocumentDB. El siguiente comando restaura los índices del clúster de Amazon DocumentDB de destino desde el directorio especificado.

```
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
```

Para confirmar que restauró los índices correctamente, conéctese al clúster de Amazon DocumentDB de destino con el intérprete de comandos de mongo y enumere los índices de una colección determinada. Consulte el siguiente código:

```
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()
```

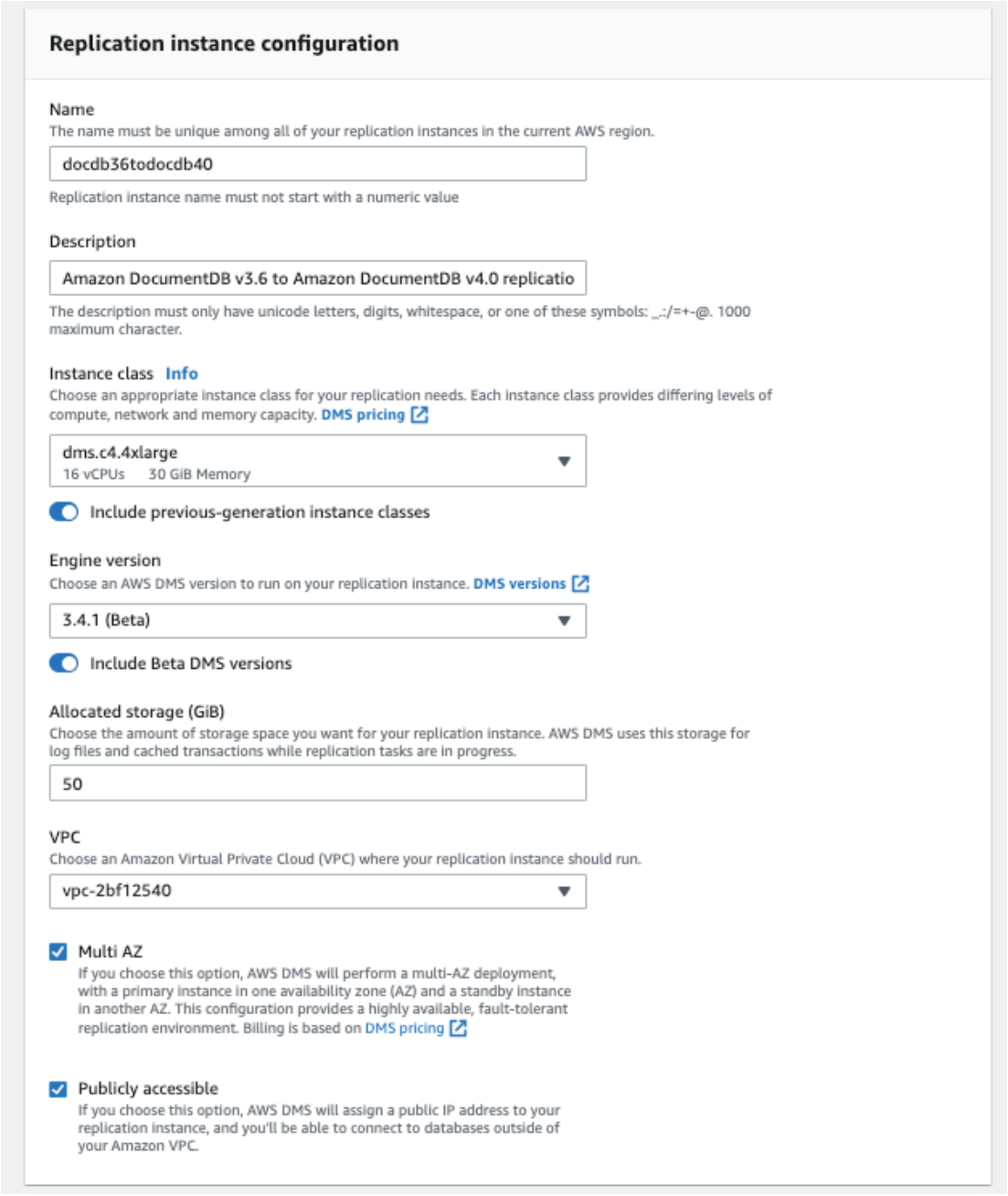
Paso 4: Cree un AWS DMS instancia de replicación

Una instancia de AWS DMS replicación conecta y lee los datos del clúster de Amazon DocumentDB de origen y los escribe en el clúster de Amazon DocumentDB de destino. La instancia de AWS DMS replicación puede realizar tanto operaciones de carga masiva como de CDC. La mayor parte de este procesamiento ocurre en la memoria. No obstante, es posible que en las operaciones de mayor volumen se precise almacenar en la memoria búfer del disco. Las transacciones almacenadas en caché y los archivos de registro también se escriben en el disco. Una vez que se migran los datos, la instancia de replicación también transmite cualquier evento de cambio para asegurarse de que el origen y el destino estén sincronizados.

Para crear una instancia AWS DMS de replicación:

1. Abra la AWS DMS consola.
2. En el panel de navegación, elija Instancias de replicación.
3. Elija Create replication instance (Crear instancia de replicación) y escriba la siguiente información:

   - En Nombre, escriba un nombre de su elección. Por ejemplo, `docdb36todocdb40`.
   - En Descripción, introduzca una descripción de su preferencia. En listitem, instancia de replicación de Amazon DocumentDB 3.6 a Amazon DocumentDB 4.0.
   - En la clase de instancia, elija el tamaño según sus necesidades.
   - En la versión Engine, elija `3.4.1.`
   - En el caso de Amazon VPC, elija la Amazon VPC que aloja los clústeres de Amazon DocumentDB de origen y destino.
   - Para el Almacenamiento asignado (GiB), utilice el valor predeterminado de 50 GiB. Si tiene una carga de trabajo de alto rendimiento de escritura, aumente este valor para que se adapte a su carga de trabajo.
   - Para Multi-AZ, elija Sí si necesita alta disponibilidad y soporte de conmutación por error.
   - En Publicly accessible (Accesible públicamente), habilite esta opción.



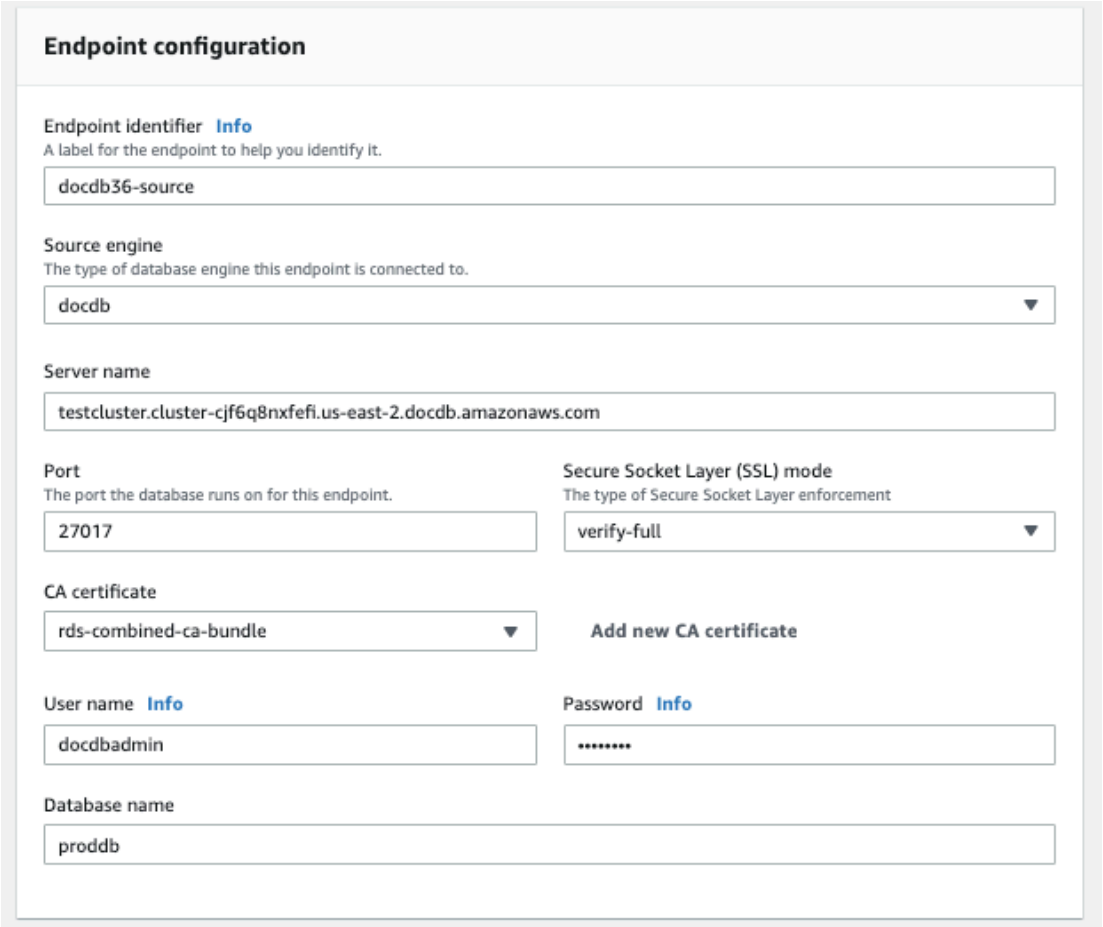
4. Elija Create replication instance.

Paso 5: Cree un AWS DMS punto final de origen

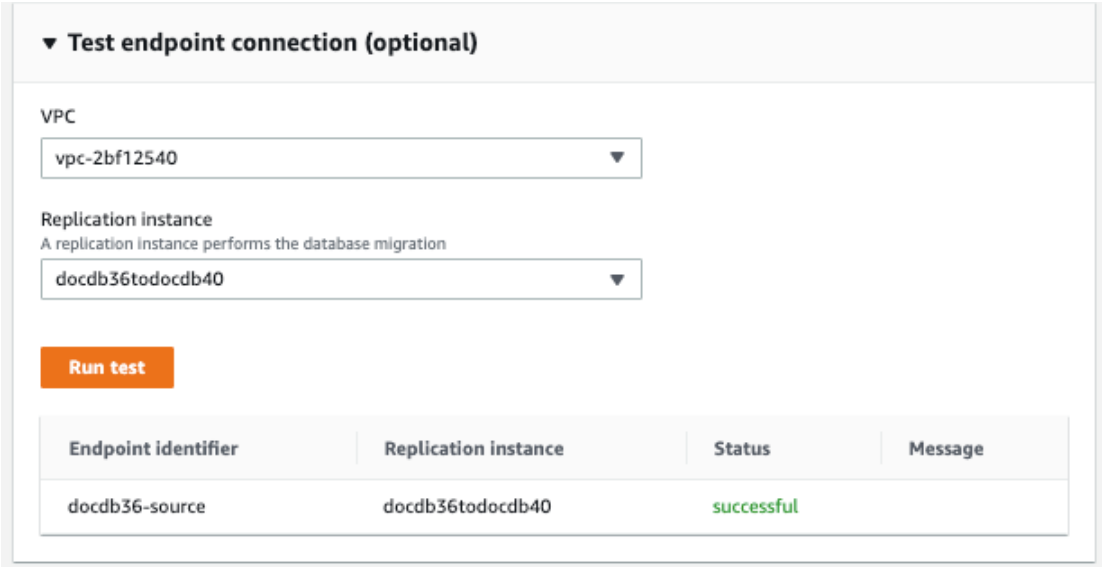
El punto de conexión de origen se utiliza para el clúster Amazon DocumentDB de origen.

Creación de un punto de conexión de origen

1. Abre la AWS DMS consola.
2. En el panel de navegación, elija Puntos de conexión.
3. Elija `Create endpoint` y especifique la siguiente información:
   - En Tipo de punto de conexión, elija Source (Origen).
   - >En Identificador de punto de conexión, escriba un nombre que sea fácil de recordar, como `docdb-source`.
   - En Motor de origen, seleccione `docdb`.
   - En Nombre de servidor, escriba el nombre DNS de su clúster de Amazon DocumentDB de origen.
   - En Puerto, escriba el número de puerto de su clúster de Amazon DocumentDB.
   - En Modo de SSL, elija `verify-full`.
   - En Certificado CA, seleccione Añadir un nuevo certificado CA. Descargue el nuevo certificado CA, el nuevo certificado CA para crear el paquete de conexiones TLS. En Identificador del certificado, escriba `rds-combined-ca-bundle` . En Importar archivo de certificado, elija Seleccionar archivo y acceda al archivo `.pem` que descargó anteriormente. Seleccione el archivo y ábralo. Elija Importar certificado y luego elija `rds-combined-ca-bundle` de Elija un certificado.
   - En Nombre de usuario, escriba el nombre de usuario principal del clúster de Amazon DocumentDB de origen.
   - En Contraseña, escriba la contraseña principal del clúster de Amazon DocumentDB de origen.
   - En Nombre de la base de datos, introduzca el nombre de la base de datos que desee actualizar.



4. Pruebe su conexión para comprobar que se configuró correctamente.



5. Seleccione Crear punto de conexión.

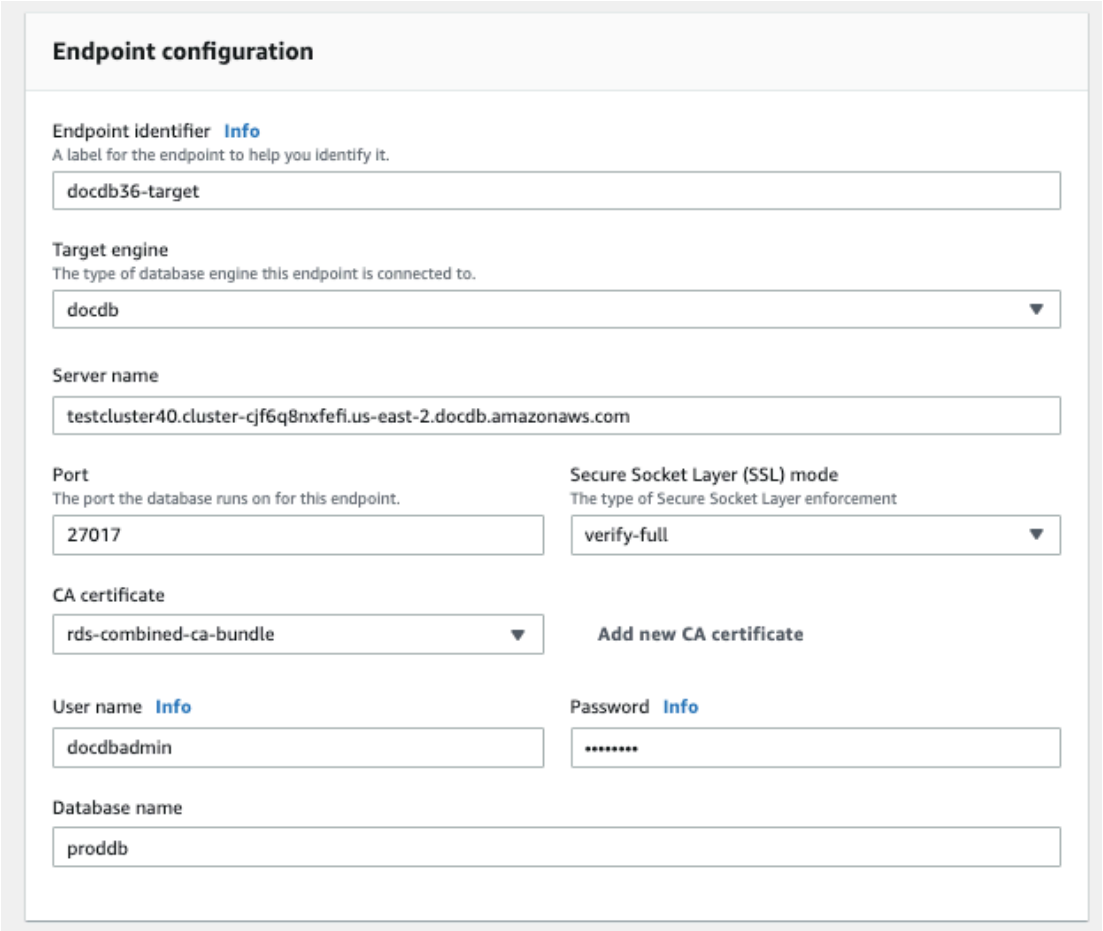
> **Note**
>
> AWS DMS solo puede migrar una base de datos a la vez.

Paso 6: Crear una AWS DMS punto final de destino

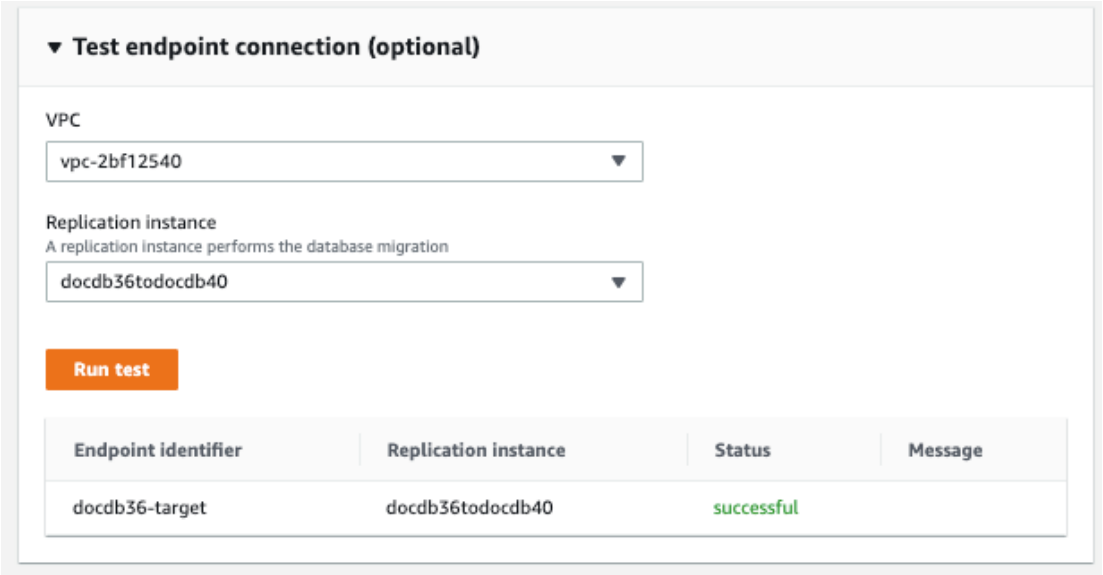
El punto de conexión de destino es para su clúster de Amazon DocumentDB de destino.

Para crear un punto de conexión de destino:

1. Abra la consola de AWS DMS.
2. En el panel de navegación, elija Puntos de conexión.
3. Elija Create endpoint (Crear punto de conexión) y escriba la siguiente información:
   - En Endpoint Type (Tipo de punto de conexión), elija Target (Destino).
   - En Endpoint identifier (identificador de punto de conexión), escriba un nombre que sea fácil de recordar, como `docdb-target`.
   - En Motor de origen, seleccione `docdb`.
   - En Nombre de servidor, escriba el nombre DNS de su clúster de Amazon DocumentDB de destino.
   - En Puerto, escriba el número de puerto de su clúster de Amazon DocumentDB de destino.
   - En Modo de SSL, elija `verify-full`.
   - En Certificado CA, elija el `rds-combined-ca-bundle` certificado existente en el menú desplegable Elija un certificado.
   - En Nombre de usuario, escriba el nombre de usuario principal del clúster de Amazon DocumentDB de destino.
   - En Contraseña, escriba la contraseña principal del clúster de Amazon DocumentDB de destino.
   - En Nombre de la base de datos, introduzca el mismo nombre de base de datos que utilizó para configurar el punto de conexión de origen.



4. Pruebe la conexión para comprobar que se configuró correctamente.



5. Elija Create Endpoint (Crear punto de conexión).

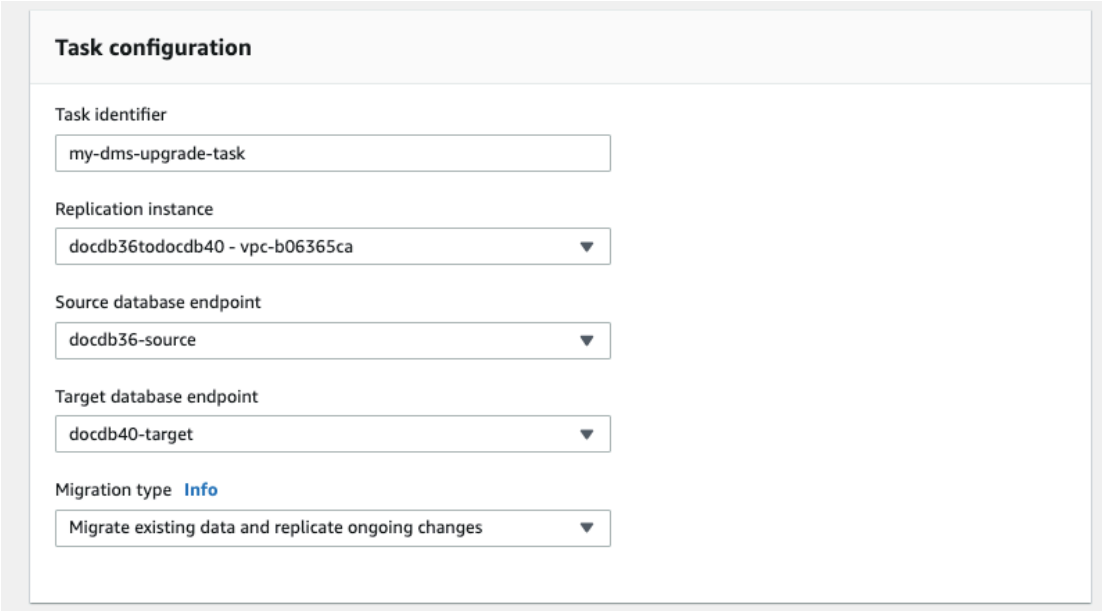
Paso 7: crear y ejecutar una tarea de migración

Una AWS DMS tarea vincula la instancia de replicación con la instancia de origen y la de destino. Al crear una tarea de migración, especifique el punto de conexión de origen, el punto de conexión de destino y la instancia de replicación, junto con cualquier configuración de migración deseada. Se puede crear una AWS DMS tarea con tres tipos de migración diferentes: migrar los datos existentes, migrar los datos existentes y replicar los cambios en curso o replicar únicamente los cambios en los datos. Debido a que el objetivo de este tutorial es actualizar un clúster de Amazon DocumentDB con un tiempo de inactividad mínimo, los pasos utilizan la opción de migrar los datos existentes y de replicar los cambios en curso. Con esta opción, AWS DMS captura los cambios mientras migra los datos existentes. AWS DMS sigue capturando y aplicando los cambios incluso después de que se hayan cargado los datos masivos. Con el tiempo, las bases de datos de origen y de destino se sincronizarán, por lo que el tiempo de inactividad de la migración será mínimo.

A continuación, se detallan los pasos para crear una tarea de migración con un tiempo de inactividad mínimo:

1. Abre la AWS DMS consola.
2. En el panel de navegación, elija Tareas de migración de base de datos.
3. Elija Crear tarea de migración de base de datos e introduzca la siguiente información en la sección Configuración de tareas:

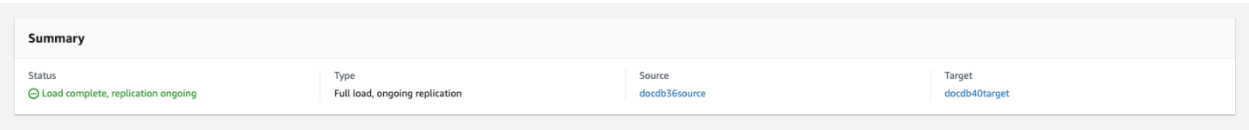
   - En Identificador de la tarea, escriba un nombre que sea fácil de recordar, como `my-dms-upgrade-task`.
   - En el Nombre de recurso de Amazon (ARN) descriptivo, introduzca un nombre fácil de usar para anular el ARN de DMS predeterminado.
   - En Replication instance (Instancia de replicación), elija la instancia de replicación que creó en Paso 4: Cree un AWS DMS instancia de replicación.
   - En Punto de conexión de la base de datos de origen, elija el punto de conexión de origen que creó en Paso 5: Cree un AWS DMS punto final de origen.
   - En Punto de conexión de destino, elija el punto de conexión de destino que creó en Paso 6: Crear una AWS DMS punto final de destino.
   - En Tipo de migración, elija Migración y réplica.



4. Introduzca la siguiente información en la sección Configuración de tareas:

   - En la sección Modo de preparación de la tabla de destino, elija No hacer nada. Esto garantizará que los índices creados en el paso 3 no se eliminen.
   - En la subsección Registros de tareas, selecciona Activar CloudWatch registros.
   - En Configuración de inicio de la tarea de migración, seleccione Automáticamente con la creación. Esto iniciará la tarea de migración automáticamente una vez que la haya creado.
   - Seleccione Crear tarea de migración de bases de datos.

AWS DMS ahora comienza a migrar los datos del clúster de Amazon DocumentDB de origen al clúster de Amazon DocumentDB de destino. El estado de la tarea debe cambiar de Iniciándose a En ejecución. Puede supervisar el progreso seleccionando Tareas en la consola. AWS DMS Después de varias minutes/hours (según el tamaño de la migración), el estado debería cambiar de Carga completa y la replicación en curso. Esto significa que AWS DMS ha completado una migración a carga completa del clúster de Amazon DocumentDB de origen a un clúster de Amazon DocumentDB de destino y ahora está replicando los eventos de cambio.



Finalmente, el origen y el destino se sincronizarán. Para comprobar si están sincronizados, ejecute una operación `count()` en sus colecciones para comprobar que todos los eventos de cambio se hayan migrado.

Paso 8: cambiar el punto de conexión de origen de la aplicación al clúster de Amazon DocumentDB de destino

Cuando se complete la carga completa y el proceso de CDC se replique de forma continua, estará listo para cambiar el punto de conexión de la base de datos de su aplicación del clúster de Amazon DocumentDB de origen al clúster de Amazon DocumentDB de destino.

Uso de una versión de soporte a largo plazo (LTS)

Amazon DocumentDB designa determinadas versiones como versiones Long-Term Support (LTS). Los clústeres de bases de datos que utilizan LTS pueden permanecer en la misma versión durante más tiempo y someterse a menos ciclos de actualización que los clústeres que utilizan versiones que no son de LTS. Las versiones LTS solo se parchearán en caso de que se introduzcan parches críticos de estabilidad y seguridad sin introducir nuevas funciones. Es posible que se publiquen parches opcionales adicionales, como métricas, para depurar errores críticos. Se aplicarán de forma acumulativa con los parches necesarios.

Versión actual de LTS

La versión LTS actual de Amazon DocumentDB es:

- Versión secundaria 5.0.0

Para determinar la versión en la que se encuentra su clúster, consultethe section called “¿Cómo comprobar si el clúster de la versión 5.0.0 de mi motor actual está en LTS”.

Para obtener más información sobre los plazos de soporte y los ciclos de lanzamiento de las versiones LTS, consulte. Fechas de soporte de la versión del motor de Amazon DocumentDB Amazon DocumentDB anunciará la próxima versión de LTS a través de los canales de comunicación estándar.

> **Note**
>
> Aún no se ha designado una versión LTS para Amazon DocumentDB 8.0. Si necesita LTS, utilice la versión 5.0.0 del motor con la versión de parche 3.0.17983 o posterior.

¿Quién debe usar LTS

Las versiones LTS de Amazon DocumentDB están diseñadas para clústeres con ventanas de actualización limitadas. El LTS es ideal para entornos de producción en los que la disponibilidad de las bases de datos afecta directamente a las operaciones empresariales.

Considere las versiones de LTS si:

- Su aplicación tiene requisitos estrictos de tiempo de actividad que limitan las oportunidades de actualización
- Cuenta con todas las funciones y mejoras necesarias del motor de base de datos que su aplicación necesita

En su lugar, considere la posibilidad de lanzar versiones estándar si:

- Su aplicación puede gestionar una interrupción mínima
- Desea disponer de las funciones y mejoras más recientes en cuanto estén disponibles

> **Important**
>
> - Las actualizaciones son operaciones unidireccionales: no se puede bajar de categoría después de la actualización
> - El clúster experimentará un breve tiempo de inactividad durante el proceso de actualización

¿Cómo crear un clúster LTS nuevo

Para crear un clúster LTS, siga estos pasos:

1. En la consola de administración, cree un clúster nuevo con la versión 5.0.0 del motor

2. Compruebe que utiliza la versión 5.0.0 del motor después de crear el clúster mediante: AWS CLI

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

La salida de esta operación será similar a lo que se indica a continuación:

```
"5.0.0"
```

Cómo actualizar de la versión 3.6.0 o 4.0.0 del motor a un clúster LTS 5.0.0

Para actualizar a un clúster LTS, siga estos pasos:

1. En la consola de administración, actualice la versión principal (MVU) del clúster de la versión 3.6.0 o 4.0.0 del motor a la 5.0.0. Para obtener instrucciones sobre cómo ejecutar una MVU, consulte. Actualización local de la versión principal Amazon DocumentDB
2. Compruebe que utiliza la versión 5.0.0 del motor después de la actualización mediante: AWS CLI

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

¿Cómo comprobar si el clúster de la versión 5.0.0 de mi motor actual está en LTS

Para comprobar si el clúster de la versión 5.0.0 de su motor ejecuta una versión LTS:

1. Compruebe que está utilizando la versión 5.0.0 del motor mediante: AWS CLI

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

2. Si es necesario, actualice el clúster actual de la versión 5.0.0 del motor. Para obtener instrucciones sobre cómo realizar una actualización de un parche, consulteActualización de un parche a la versión del motor de un clúster.

Soporte extendido de Amazon DocumentDB

El Soporte extendido de Amazon DocumentDB le permite seguir ejecutando una versión del motor después de la fecha de finalización del soporte estándar por un costo adicional. Los clústeres se inscriben en el Soporte extendido una vez finalizada la fecha del soporte estándar. Si no completa las actualizaciones antes de que finalice la fecha de soporte estándar de la versión del motor, los clústeres se facturarán según la tarifa del Soporte extendido.

Temas

- Información general del soporte extendido
- Precios del Soporte extendido
- Responsabilidades

El Soporte extendido de Amazon DocumentDB ofrece las siguientes actualizaciones y soporte técnico:

- Actualizaciones de seguridad para las CVE críticas y altas para la instancia o el clúster
- Correcciones de errores y parches para problemas críticos
- La posibilidad de abrir casos de soporte y recibir ayuda para la solución de problemas según el Acuerdo de nivel de servicio de Amazon DocumentDB.

Esta oferta de pago le da más tiempo para llevar a cabo la actualización a una versión del motor compatible. Por ejemplo, la fecha de finalización del soporte estándar de Amazon DocumentDB para la versión 3.6 es el 30 de marzo de 2026; sin embargo, no estará preparado para actualizar a la versión 5.0 u 8.0 antes de esa fecha. En este caso, Amazon DocumentDB inscribirá su clúster de la versión 3.6 en soporte extendido el 31 de marzo de 2026 para que pueda seguir utilizando la versión 3.6 de Amazon DocumentDB. A partir del 1 de julio de 2026, le AWS cobrará una prima de soporte extendido para los clústeres de la versión 3.6 de Amazon DocumentDB.

El Soporte extendido de Amazon DocumentDB está disponible durante un máximo de 3 años después de la fecha de finalización del soporte estándar para la versión del motor. Transcurrido este tiempo, si no ha actualizado la versión del motor a una versión compatible, Amazon DocumentDB actualizará la versión del motor. Actualice a una versión de motor compatible lo antes posible.

Fecha de finalización del soporte estándar de la versión 3.6 de Amazon DocumentDB y fechas del Soporte extendido:

- Publicación de la versión 3.6: 9 de enero de 2019
- Finalización del soporte estándar de la versión 3.6: 30 de marzo de 2026
- Inicio de la versión 3.6: precios del primer año de soporte extendido: 1 de julio de 2026
- Inicio del Soporte extendido de la versión 3.6 (precio del tercer año): 31 de marzo de 2028
- Finalización del Soporte extendido de la versión 3.6: 30 de marzo de 2029

Información general del soporte extendido

Amazon DocumentDB actualiza el clúster a la última versión del motor publicada (antes de que finalice la fecha del Soporte extendido), si el clúster aún no está en esa versión. El período de actualización no tendrá lugar hasta después de la fecha de soporte estándar para esa versión del motor.

Puede crear nuevos clústeres con una versión del motor que haya alcanzado la fecha de finalización del soporte estándar, si ya tiene un clúster en esa versión del motor. Amazon DocumentDB inscribe estos nuevos clústeres en el Soporte extendido de Amazon DocumentDB y le cobra por esta oferta. Si actualiza a una versión del motor que aún se encuentra recibiendo el soporte estándar de Amazon DocumentDB antes de la fecha de finalización del soporte estándar de la versión 3.6, no se le cobrará la prima del Soporte extendido.

Puede finalizar la inscripción en el Soporte extendido de Amazon DocumentDB en cualquier momento. Para finalizar la inscripción, actualice cada clúster inscrito a una versión de motor más reciente cuyo soporte estándar de Amazon DocumentDB siga vigente. La finalización de la inscripción en el Soporte extendido de Amazon DocumentDB entrará en vigor el día en que complete una actualización a una versión más reciente del motor cuyo soporte estándar de Amazon DocumentDB siga vigente.

Para obtener más información sobre las fechas de finalización del soporte estándar de Amazon DocumentDB y las fechas de finalización del Soporte extendido de Amazon DocumentDB, consulte Fechas de soporte de la versión del motor de Amazon DocumentDB.

Precios del Soporte extendido

A partir del 1 de julio de 2026, se cobrarán cargos por todos los clústeres que cuenten con soporte extendido. Para conocer la fecha de finalización del soporte estándar de la versión 3.6 de Amazon DocumentDB, consulte Fechas de soporte de la versión del motor de Amazon DocumentDB.

El cargo adicional del Soporte extendido de Amazon DocumentDB se detiene cuando realiza una de las siguientes acciones:

- Actualiza a una versión de motor incluida en el soporte estándar.
- Elimina el clúster que se ejecuta en la versión 3.6 pasada la fecha de finalización del soporte estándar.

Por ejemplo, la versión 3.6 de Amazon DocumentDB entrará en servicio de soporte extendido el 31 de marzo de 2026, pero los cargos no comenzarán hasta el 1 de julio de 2026. Si actualiza su clúster de la versión 3.6 de Amazon DocumentDB a la versión 5.0 de Amazon DocumentDB el 31 de julio de 2026, solo se le cobrarán 30 días de soporte extendido en la versión 3.6 de Amazon DocumentDB. Para obtener más información, consulte Precios de Amazon DocumentDB.

Responsabilidades

Las siguientes son las responsabilidades de Amazon DocumentDB y sus responsabilidades en relación con el Soporte extendido de Amazon DocumentDB.

Responsabilidades de Amazon DocumentDB

Tras la fecha de finalización del soporte estándar de la versión 3.6 de Amazon DocumentDB, Amazon DocumentDB aplicará parches, correcciones de errores y actualizaciones a los clústeres de la versión 3.6 inscritos en el Soporte extendido.

Sus responsabilidades

Usted es responsable de actualizar el motor a una versión más reciente antes de la fecha de finalización del Soporte extendido de la versión 3.6 de Amazon DocumentDB. Si no actualiza el clúster, después de la fecha de finalización del Soporte extendido de la versión 3.6 de Amazon DocumentDB, este intentará actualizar el clúster a la última versión del motor con soporte estándar de Amazon DocumentDB.

Características y configuraciones de Amazon DocumentDB

En las tablas siguientes se comparan las características, las capacidades y los tipos de instancias compatibles entre las versiones del motor de Amazon DocumentDB y los clústeres elásticos. Utilice estas tablas para determinar qué versión del motor se adapta mejor a sus requisitos de carga de trabajo.

Estado del motor

Estado del motor por versión

Característica/capacidad	v3.6	v4.0	v5.0	v8.0	
Estado de soporte	EOL: marzo de 2026	Activo	Activo (LTS disponible)	Activo (Georgia, noviembre de 2025)	
Soporte extendido disponible	Sí (de pago, hasta marzo de 2029)	N/A	N/A	N/A	
Versión LTS disponible	No	No	Sí (v5.0 LTS)	No	
Clústeres elásticos	No	No	Sí	No	

Tipos de clústeres

Tipos de clústeres por versión

Cluster Type (Tipo de clúster)	v3.6	v4.0	v5.0	v8.0	Clústeres elásticos
Instance-based clúster	Sí	Sí	Sí	Sí	Sí

Cluster Type (Tipo de clúster)	v3.6	v4.0	v5.0	v8.0	Clústeres elásticos
instancias sin servidor	No	No	Sí	Sí	No

Características

Características y configuraciones por versión del motor

Característica	v3.6	v4.0	v5.0	v8.0	Clústeres elásticos
Clústeres globales	No	Sí	Sí	Sí	No
Multi-AZ (3 A)	Sí	Sí	Sí	Sí	Sí (por fragmento)
Se admiten réplicas de lectura	15	15	15	15	15 por fragmento
Transacciones ACID	No	Sí	Sí	Sí	No
Reintento de las escrituras	No	No	No	Sí (8.0.2+)	No
Change streams	Sí	Sí	Sí	Sí	No
Búsqueda vectorial	No	No	Sí	Sí	No
Información de rendimiento	Sí	Sí	Sí	Sí	No
In-place actualización de la versión principal (MVU)	Sí (a 5.0 o 8.0)	Sí (a 5.0 o 8.0)	Sí (a 8.0)	No (solo objetivo)	No
Client-side cifrado a nivel de campo (FLE)	No	No	Sí	Sí	No

Característica	v3.6	v4.0	v5.0	v8.0	Clústeres elásticos
I/O-Optimized almacenamiento	No	No	Sí	Sí	No
Rotación del certificado TLS (sin reinicio)	Sí (parche 1.0.208662+)	Sí (parche 2.0.10179+)	Sí (parche 3.0.4780+)	Sí	Sí
Cross-Region copia instantánea	Sí	Sí	Sí	Sí	Sí
Cross-account compartir instantáneas	Sí	Sí	Sí	Sí	No
Registros de auditoría y consultas lentas	Sí	Sí	Sí	Sí	Sí
Dual-stack IPv4/IPv6	No	Sí (parche 2.0.11747+)	Sí (parche 3.0.15902+)	Sí	N/A
Collation (Intercalación)	No	No	No	Sí	No
Puntos de vista	No	No	No	Sí	No
Compatibilidad con la compresión	No	No	LZ4	LZ4/Ztsd	No
Índice de texto	No	No	V1	V2	No
FedRamp () GovCloud	No	No	Sí	Sí	No

Tipos de instancias

Soporte de tipos de instancia por versión

Tipo de instancia	v3.6	v4.0	v5.0	v8.0	
db.r4 (generación anterior)	Sí	No	No	No	
db.r5	Sí	Sí	Sí	Sí	
db.r6g (Graviton 2)	No	Sí	Sí	Sí	
db.r6gd (NVMe-backed Gravitón 2)	No	No	Sí	Sí	
db.r8g (Gravitón 4)	No	No	Sí	Sí	
db.t3.medium	Sí	Sí	Sí	Sí	
db.t4g.medium (Gravitón 2)	No	Sí	Sí	Sí	
sin servidor	No	No	Sí	Sí	

AWS integraciones

AWS soporte de integración por versión

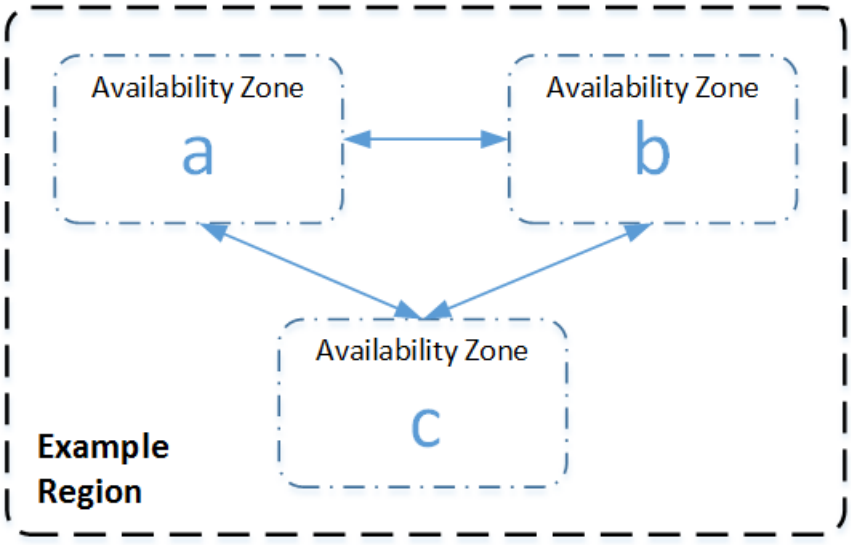
AWS Integración//Función	v3.6	v4.0	v5.0	v8.0	Clústeres elásticos
AWS Backup	Sí	Sí	Sí	Sí	No
AWS Secrets Manager	Sí	Sí	Sí	Sí	No
AWS Lambda (ESM nativo)	No	Sí	Sí	No	No

AWS Integración//Función	v3.6	v4.0	v5.0	v8.0	Clústeres elásticos
Ingestión OpenSearch de servicios de Amazon () Zero-ETL	No	Sí	Sí	Sí	No
AWS DMS soporte de migración	Sí	Sí	Sí	Sí (a partir de la versión 5.0 AWS DMS)	Sí
AWS Identity and Access Management authentication	No	No	Sí (basado en la instancia 5.0)	Sí	No
AWS CloudShell	No	No	Sí	Sí	No
AWS Key Management Service	Sí	Sí	Sí	Sí	Sí
AWS CloudTrail	Sí	Sí	Sí	Sí	Sí

Elección de regiones y zonas de disponibilidad

Los recursos de computación en la nube de Amazon están alojados en varias ubicaciones de todo el mundo. Estas ubicaciones constan Regiones de AWS de zonas de disponibilidad. Cada Región de AWS es un área geográfica independiente. Cada región tiene varias ubicaciones aisladas conocidas como zonas de disponibilidad. Amazon DocumentDB le ofrece la posibilidad de colocar recursos, como instancias, y datos en varias ubicaciones. Los recursos no se replican en todos los demás, Regiones de AWS a menos que lo haga específicamente.

Amazon opera centros de datos de alta disponibilidad con tecnología de vanguardia. Aunque es infrecuente, puede suceder que se produzcan errores que afecten a la disponibilidad de las instancias que están en la misma ubicación. Si hospeda todas las instancias en una misma ubicación y se produce un error en ella, ninguna de las instancias estaría disponible. El siguiente diagrama muestra una Región de AWS con tres zonas de disponibilidad.



Es importante recordar que cada región es completamente independiente. Cualquier actividad de Amazon DocumentDB que se inicie (por ejemplo, la creación de instancias o la enumeración de las instancias de base de datos disponibles) se ejecuta solo en la Región de AWS predeterminada actual. Puede cambiar la región predeterminada en la consola configurando la variable de entorno `EC2_REGION`. También puede invalidarla mediante el parámetro `--region` de la AWS CLI. Para obtener más información, consulte Configurar AWS Command Line Interface, específicamente, las secciones sobre variables de entorno y opciones de línea de comandos.

Cuando crea un clúster mediante la consola de Amazon DocumentDB y decide crear una réplica en una zona de disponibilidad diferente, Amazon DocumentDB crea dos instancias. Crea la instancia principal en una zona de disponibilidad y la instancia de réplica en una zona de disponibilidad diferente. El volumen del clúster siempre se replica en tres zonas de disponibilidad.

Para crear o trabajar con una instancia de Amazon DocumentDB en una instancia específica Región de AWS, utilice el punto de enlace de servicio regional correspondiente.

Disponibilidad por región

Amazon DocumentDB está disponible en las siguientes AWS regiones.

Regiones compatibles con Amazon DocumentDB

Nombre de la región	Region	Zonas de disponibilidad (cálculo)
Este de EE. UU. (Ohio)	`us-east-2`	3

Nombre de la región	Region	Zonas de disponibilidad (cálculo)
Este de EE. UU. (Norte de Virginia)	us-east-1	6
Oeste de EE. UU. (Oregón)	us-west-2	4
África (Ciudad del Cabo)	af-south-1	3
América del Sur (São Paulo)	sa-east-1	3
Asia-Pacífico (Hong Kong)	ap-east-1	3
Asia-Pacífico (Hyderabad)	ap-south-2	3
Asia-Pacífico (Malasia)	ap-southeast-5	3
Asia-Pacífico (Mumbai)	ap-south-1	3
Asia-Pacífico (Osaka)	ap-northeast-3	3
Asia-Pacífico (Seúl)	ap-northeast-2	4
Asia-Pacífico (Singapur)	ap-southeast-1	3
Asia-Pacífico (Sídney)	ap-southeast-2	3
Asia-Pacífico (Yakarta)	ap-southeast-3	3
Asia-Pacífico (Melbourne)	ap-southeast-4	3

Nombre de la región	Region	Zonas de disponibilidad (cálculo)
Asia-Pacífico (Tailandia)	ap-southeast-7	3
Asia-Pacífico (Tokio)	ap-northeast-1	3
Canadá (centro)	ca-central-1	3
Oeste de Canadá (Calgary)	ca-west-1	3
Región China (Pekín)	cn-north-1	3
China (Ningxia)	cn-northwest-1	3
Europa (Fráncfort)	eu-central-1	3
Europa (Zúrich)	eu-central-2	3
Europa (Irlanda)	eu-west-1	3
Europa (Londres)	eu-west-2	3
Europa (Milán)	eu-south-1	3
Europa (París)	eu-west-3	3
Europa (España)	eu-south-2	3
Europa (Estocolmo)	eu-north-1	3
México (centro)	mx-central-1	3
Medio Oriente (EAU)	me-central-1	3
Israel (Tel Aviv)	il-central-1	3
AWS GovCloud (US-West)	us-gov-west-1	3

Nombre de la región	Region	Zonas de disponibilidad (cálculo)
AWS GovCloud (US-East)	`us-gov-east-1`	3

Por defecto, la zona horaria de un clúster de Amazon DocumentDB es la hora universal coordinada (UTC).

Para obtener información sobre cómo encontrar los puntos de conexión para los clústeres e instancias en una región determinada, consulte. Descripción de los puntos de conexión de Amazon DocumentDB

Zero-ETL integración con Amazon OpenSearch Service

Temas

- Amazon OpenSearch Service como destino
- Limitaciones

Amazon OpenSearch Service como destino

OpenSearch La integración de los servicios con Amazon DocumentDB le permite transmitir a los OpenSearch dominios eventos de carga completa y cambios de datos. La infraestructura de ingesta se aloja como canalizaciones de OpenSearch ingesta y proporciona un mecanismo de alta escala y baja latencia para transmitir datos de forma continua desde las colecciones de Amazon DocumentDB.

Durante la carga completa, la integración sin ETL extrae primero los datos históricos de carga completa para utilizarlos en una canalización de ingesta. OpenSearch Una vez ingeridos los datos a plena carga, las canalizaciones de OpenSearch ingesta comenzarán a leer los datos de los flujos de cambios de Amazon DocumentDB y, finalmente, se pondrán al día para mantener la coherencia de los datos casi en tiempo real entre Amazon DocumentDB y. OpenSearch OpenSearch almacena los documentos en índices. Los datos entrantes de una colección de Amazon DocumentDB se pueden enviar a un índice o se pueden dividir en índices diferentes. Las canalizaciones de ingesta sincronizan todos los eventos de creación, actualización y eliminación de una colección de Amazon DocumentDB a la vez que crean, actualizan y eliminan OpenSearch documentos para mantener sincronizados ambos sistemas de datos. Los canales de ingesta se pueden configurar para

leer datos de una colección y escribirlos en un índice, o leer datos de una colección y enviarlos condicionalmente a varios índices.

Las canalizaciones de ingesta se pueden configurar para transmitir datos desde Amazon DocumentDB a Amazon Service mediante: OpenSearch

- Solo carga completa
- Transmisión de eventos de flujos de cambios desde Amazon DocumentDB sin carga completa
- Carga completa seguida de flujos de cambios desde Amazon DocumentDB

Para configurar la canalización de ingesta, siga estos pasos:

Paso 1: Cree un dominio de Amazon OpenSearch Service o una colección sin servidor OpenSearch

Se requiere una colección de Amazon OpenSearch Service con los permisos adecuados para leer los datos. Consulte Introducción a Amazon OpenSearch Service o Introducción a Amazon OpenSearch Serverless en la Guía para desarrolladores de Amazon OpenSearch Service para crear una colección. Consulte Amazon OpenSearch Ingestion en la Guía para desarrolladores de Amazon OpenSearch Service para crear un rol de AIM con los permisos correctos para acceder a los datos de escritura en la colección o el dominio.

Paso 2: habilitación de los flujos de cambios en el clúster de Amazon DocumentDB

Asegúrese de que los flujos de cambios estén habilitados en las colecciones requeridas del clúster de Amazon DocumentDB. Para obtener más información, consulte Uso de flujos de cambios con Amazon DocumentDB.

Paso 3: configuración del rol de canalización con permisos para escribir en el bucket de Amazon S3 y en el dominio o colección de destino

Una vez creada la colección de Amazon DocumentDB y habilitado el flujo de cambios, configure el rol de canalización que desee usar en la configuración de canalización y agregue los siguientes permisos al rol:

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/*"
            ]
        }
    ]
}
```

Para que una OpenSearch canalización escriba datos en un OpenSearch dominio, el dominio debe tener una política de acceso a nivel de dominio que permita al rol de canalización sts_role_arn acceder a él. El siguiente ejemplo de política de acceso al dominio permite que el rol de canalización denominado `pipeline-role`, que creó en el paso anterior, escriba datos en el dominio denominado `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}/*"
    }
  ]
}
```

Paso 4: Añade los permisos necesarios para crear el rol de canalización 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" }
            }
        }
    ]
```

```
}
```

Paso 5: creación de la canalización

Configure una canalización de OpenSearch ingesta especificando Amazon DocumentDB como fuente. En este ejemplo de configuración de canalización se presupone el uso de un mecanismo de obtención del flujo de cambios. Para obtener más información, consulte Uso OpenSearch de una canalización de ingesta con Amazon DocumentDB en la Guía para desarrolladores OpenSearch de Amazon Service.

Limitaciones

Se aplican las siguientes limitaciones a la integración de Amazon OpenSearch DocumentDB:

- Solo se admite una colección de Amazon DocumentDB como origen por canalización.
- Cross-region no se admite la ingesta de datos. El clúster y el OpenSearch dominio de Amazon DocumentDB deben estar en el mismo lugar. Región de AWS
- Cross-account no se admite la ingesta de datos. El clúster y la canalización de OpenSearch ingesta de Amazon DocumentDB deben estar en la misma cuenta. AWS
- No se admiten los clústeres elásticos de Amazon DocumentDB. Solo se admiten los clústeres basados en instancias de Amazon DocumentDB.
- Asegúrese de que el clúster de Amazon DocumentDB tenga habilitada la autenticación mediante secretos. AWS AWS los secretos son el único mecanismo de autenticación admitido.
- La configuración de canalización existente no se puede actualizar para incorporar datos de una base de datos diferente o de and/or una colección diferente. Para actualizar el nombre de la and/or colección de bases de datos de una canalización, debe crear una canalización nueva.

Desarrollo de Amazon DocumentDB

En esta sección se describe la conexión a su clúster de Amazon DocumentDB, la consulta y la administración de datos, el trabajo con índices y el uso de funciones como los flujos de cambios, las transacciones y la búsqueda vectorial para aplicaciones de IA generativa.

Temas

- Conexión a Amazon DocumentDB
- Programación con Amazon DocumentDB
- Consultas en Amazon DocumentDB
- Índices
- Uso de flujos de cambios con Amazon DocumentDB
- Transacciones en Amazon DocumentDB
- Escrituras que se pueden volver a intentar en Amazon DocumentDB
- Uso de la intercalación en Amazon DocumentDB 8.0
- Uso de vistas en Amazon DocumentDB 8.0
- Inteligencia artificial generativa de Amazon DocumentDB
- Utilización AWS Lambda con flujos de cambio

Conexión a Amazon DocumentDB

Amazon DocumentDB ofrece varias opciones de conexión, incluidas las conexiones programáticas mediante controladores MongoDB, la conexión desde fuera de una VPC mediante túneles SSH o una VPN, y herramientas populares como Studio 3T y. AWS DataGrip El servicio admite conexiones de conjuntos de réplicas para una alta disponibilidad. Para el análisis y la elaboración de informes, puede conectarse mediante controladores JDBC u ODBC. Las instancias de EC2 de la misma VPC pueden conectarse directamente a los clústeres de DocumentDB. Todas AWS credentials/ authentication las conexiones requieren TLS y son adecuadas.

Temas

- Conexión mediante programación a Amazon DocumentDB
- Conexión a un clúster de Amazon DocumentDB desde fuera de una Amazon VPC
- Conexión a Amazon DocumentDB como conjunto de réplicas

- Conexión a un clúster de Amazon DocumentDB desde Studio 3T
- Connect a Amazon DocumentDB mediante DataGrip
- Conectarse mediante Amazon EC2
- Conexión mediante el controlador JDBC de Amazon DocumentDB
- Conexión mediante el controlador ODBC de Amazon DocumentDB

Conexión mediante programación a Amazon DocumentDB

En esta sección, se incluyen ejemplos de código que demuestran cómo conectarse a Amazon DocumentDB (compatible con MongoDB) utilizando diferentes lenguajes. Los ejemplos se dividen en dos secciones en función de si se conecta a un clúster que tiene habilitada o deshabilitada la seguridad de la capa de transporte (TLS). De manera predeterminada, TLS está habilitado para los nuevos clústeres de Amazon DocumentDB. Sin embargo, puede desactivar TLS si lo desea. Para obtener más información, consulte Cifrado de datos en tránsito.

Si está intentando conectarse a Amazon DocumentDB desde fuera de la VPC en la que reside el clúster, consulte. Conexión a un clúster de Amazon DocumentDB desde fuera de una Amazon VPC

Antes de conectarse a su clúster, debe saber si TLS está habilitado en él. En la siguiente sección, se muestra cómo determinar el valor del parámetro `tls` del clúster mediante la Consola de administración de AWS o la AWS CLI. A continuación, puede buscar y aplicar el ejemplo de código adecuado.

Temas

- Determinar el valor de su parámetro tls
- Conexión con TLS habilitado
- Conexión con TLS deshabilitado
- Compresión de red en Amazon DocumentDB

Determinar el valor de su parámetro tls

Determinar si su clúster tiene TLS habilitado es un proceso de dos pasos que puede realizar mediante el comando o. Consola de administración de AWS AWS CLI

1. Determine qué grupo de parámetros rige el clúster.

Using the Consola de administración de AWS

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación izquierdo, elija Clusters (Clústeres).
3. En la lista de clústeres, seleccione el nombre del clúster.
4. La página resultante muestra los detalles del clúster seleccionado. Seleccione la pestaña Configuración. En la sección Configuraciones y estado, busque el nombre del grupo de parámetros debajo del Grupo de parámetros del clúster.

Using the AWS CLI

El siguiente AWS CLI código determina qué parámetro rige su clúster. Asegúrese de reemplazar *sample-cluster* por el nombre del clúster.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

La salida de esta operación será similar a lo que se indica a continuación:

```
[
       [
           "sample-cluster",
           "sample-parameter-group"
       ]
]
```

2. Determine el valor del parámetro **`tls`** en el grupo de parámetros del clúster.

Using the Consola de administración de AWS

1. En el panel de navegación, seleccione Grupos de parámetros.
2. En la ventana Grupos de parámetros del clúster, seleccione el nombre del grupo de parámetros del clúster del Paso 1d.
3. En la página resultante se muestran los parámetros del grupo de parámetros del clúster. Puede ver el valor del parámetro `tls` aquí. Para obtener información sobre la

modificación de este parámetro, consulte Modificación de grupos de parámetros de clúster de Amazon DocumentDB.

Using the AWS CLI

Puede usar el `describe-db-cluster-parameters` AWS CLI comando para ver los detalles de los parámetros del grupo de parámetros de su clúster.

- **`--describe-db-cluster-parameters`**: para obtener una lista de los parámetros de un grupo de parámetros, junto con sus valores.
  - **`--db-cluster-parameter-group name`**: obligatorio. El nombre del grupo de parámetros del clúster.

En los siguientes ejemplos, reemplaza cada uno de *`user input placeholder`* ellos por la información de tu clúster.

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

La salida de esta operación será similar a lo que se indica a continuación:

```
{
        "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 admite los puntos de enlace FIPS 140-3 a partir de los clústeres de Amazon DocumentDB 5.0 (versión 3.0.3727 del motor) en las siguientes regiones: ca-central-1, us-west-2, us-gov-west-1.

Después de determinar el valor del parámetro `tls`, continúe conectándose al clúster utilizando uno de los ejemplos de código que se muestran en las siguientes secciones.

- Conexión con TLS habilitado
- Conexión con TLS deshabilitado

Conexión con TLS habilitado

Para ver un ejemplo de código para conectarse mediante programación a un clúster de Amazon DocumentDB, elija la pestaña correspondiente al idioma que desee utilizar. TLS-enabled

Para cifrar datos en tránsito, descargue la clave pública para Amazon DocumentDB denominada `global-bundle.pem` mediante la siguiente operación.

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

Si l aplicación está en Microsoft Windows y requiere un archivo PKCS7, puede descargar el paquete de certificados PKCS7. Este paquete contiene los certificados intermedios y raíz en. https://truststore.pki.rds.amazonaws.com/global/global-bundle.p7b

Python

El código siguiente demuestra cómo conectarse a Amazon DocumentDB utilizando Python cuando TLS está habilitado.

En el siguiente ejemplo, sustituya cada uno de *user input placeholder* ellos por la información de su clúster.

```
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

El siguiente código muestra cómo conectarse a Amazon DocumentDB Node.js cuando TLS está habilitado.

> **Important**
>
> Existe una limitación conocida en relación con Node.js los controladores anteriores a la versión 6.13.1, que actualmente no son compatibles con la autenticación de identidad de IAM en Amazon DocumentDB. Node.js Los controladores y las herramientas que utilizan el Node.js controlador (por ejemplo, mongosh) deben actualizarse para que utilicen Node.js la versión 6.13.1 o superior del controlador.

En el siguiente ejemplo, sustituya cada uno por la información de su *user input placeholder* clúster.

```
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

El código siguiente demuestra cómo conectarse a Amazon DocumentDB utilizando PHP cuando TLS está habilitado.

En el siguiente ejemplo, reemplaza cada una de *user input placeholder* ellas por la información de tu clúster.

```
<?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

El código siguiente demuestra cómo conectarse a Amazon DocumentDB utilizando Go cuando TLS está habilitado.

> **Note**
>
> A partir de la versión 1.2.1, el controlador de MongoDB Go solo utilizará el primer certificado de servidor de CA que se encuentre en `sslcertificateauthorityfile`. El siguiente código de ejemplo corrige esta limitación anexando manualmente todos los certificados de servidor que se encuentran en `sslcertificateauthorityfile` a una configuración TLS personalizada utilizada durante la creación del cliente.

En los siguientes ejemplos, sustituya cada uno *`user input placeholder`* por la información de su clúster.

```
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

Al conectarse a un clúster de TLS-enabled Amazon DocumentDB desde una aplicación Java, el programa debe usar el archivo AWS de autoridad de certificación (CA) proporcionado para validar la conexión. Para utilizar el certificado CA de Amazon RDS, haga lo siguiente:

1. Descargue el archivo de CA de Amazon RDS desde https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem .
2. Cree un almacén de confianza con el certificado CA contenido en el archivo utilizando los siguientes comandos. Asegúrese de cambiarlo por *truststore-password* otro. Si tiene acceso a un almacén de confianza que contiene el certificado CA antiguo (`rds-ca-2015-root.pem`) y el nuevo certificado CA (`rds-ca-2019-root.pem`), puede importar el grupo de certificados en el almacén de confianza.

   A continuación se muestra un ejemplo de script de intérprete de comandos que importa el paquete de certificados a un almacén de confianza en un sistema operativo Linux. En los siguientes ejemplos, sustituya cada uno *user input placeholder* por su propia información. En particular, siempre que se encuentre el directorio de ejemplo *mydir* "" en el script, sustitúyalo por un directorio que haya creado para esta tarea.

   ```
   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
```

A continuación se muestra un ejemplo de script de intérprete de comandos que importa el paquete de certificados a un almacén de confianza en macOS.

```
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. Utilice el `keystore` de su programa estableciendo las siguientes propiedades del sistema en la aplicación antes de establecer una conexión con el clúster de Amazon DocumentDB.

```
javax.net.ssl.trustStore: truststore
javax.net.ssl.trustStorePassword: truststore-password;
```

4. El código siguiente demuestra cómo conectarse a Amazon DocumentDB utilizando Java cuando TLS está habilitado.

   En el siguiente ejemplo, sustituya cada uno *user input placeholder* por la información de su clúster.

```
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

El código siguiente demuestra cómo conectarse a Amazon DocumentDB utilizando C# / .NET cuando TLS está habilitado.

En el siguiente ejemplo, reemplaza cada una de *user input placeholder* ellas por la información de tu clúster.

```
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

El siguiente código demuestra cómo conectarse a Amazon DocumentDB y realizar consultas mediante la versión más reciente, mongosh, o la versión anterior del intérprete de comandos de mongo cuando TLS está habilitado.

Conexión con Amazon DocumentDB con mongosh

> **Important**
>
> Existe una limitación conocida en relación con Node.js los controladores anteriores a la versión 6.13.1, que actualmente no son compatibles con la autenticación de identidad de IAM en Amazon DocumentDB. Node.js Los controladores y las herramientas que utilizan el Node.js controlador (por ejemplo, mongosh) deben actualizarse para que utilicen Node.js la versión 6.13.1 o superior del controlador.

En los siguientes ejemplos, sustituya cada uno de ellos por la información *`user input placeholder`* de su clúster.

```
mongosh --tls --host cluster-end-point:27017 --tlsCAFile  global-bundle.pem --
username sample-user --password password --retryWrites false
```

Conexión con Amazon DocumentDB con la versión anterior del intérprete de comandos de mongo

Si usa IAM, debe usar una versión anterior del intérprete de comandos de mongo. Introduzca una de las siguientes opciones de comando:

```
mongo --ssl --host cluster-end-point:27017 --sslCAFile global-bundle.pem --
username sample-user --password password
```

Si utiliza una versión igual o superior a la 4.2, utilice el siguiente código para conectarse. A partir de la versión 8.0.2 del motor, Amazon DocumentDB admite escrituras que se pueden volver a intentar. En las versiones anteriores del motor, no se admiten escrituras que se puedan volver a intentar. Si utiliza el intérprete de comandos de mongo heredado (no mongosh), no incluya el comando `retryWrites=false` en ninguna cadena de código. Para obtener más información, consulte Escrituras que se pueden volver a intentar en Amazon DocumentDB.

```
mongo --tls --host cluster-end-point:27017 --tlsCAFile global-bundle.pem --
username sample-user --password password
```

Pruebe la conexión

1. Insertar un documento.

   ```
   db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
   ```

2. Busque el documento que ha insertado anteriormente.

   ```
   db.myTestCollection.find({'hello':'Amazon DocumentDB'})
   ```

R

En el siguiente código, se muestra cómo conectarse a Amazon DocumentDB con R usando mongolite (https://jeroen.github.io/mongolite/) cuando TLS está habilitado.

En el siguiente ejemplo, sustituya cada una de *user input placeholder* ellas por la información de su clúster.

```
#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

El código siguiente demuestra cómo conectarse a Amazon DocumentDB con Ruby cuando TLS está habilitado.

En el siguiente ejemplo, reemplaza cada una de *user input placeholder* ellas por la información de tu clúster.

```
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
```

Conexión con TLS deshabilitado

Para ver un ejemplo de código para conectarse mediante programación a un clúster de TLS-disabled Amazon DocumentDB, seleccione la pestaña correspondiente al idioma que desee utilizar.

Python

El código siguiente demuestra cómo conectarse a Amazon DocumentDB utilizando Python cuando TLS está deshabilitado.

En el siguiente ejemplo, sustituya cada una de ellas por la *user input placeholder* información de su clúster.

```
## 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

El siguiente código muestra cómo conectarse a Amazon DocumentDB Node.js cuando TLS está deshabilitado.

> **Important**
>
> Existe una limitación conocida en relación con Node.js los controladores anteriores a la versión 6.13.1, que actualmente no son compatibles con la autenticación de identidad de IAM en Amazon DocumentDB. Node.js Los controladores y las herramientas que utilizan el Node.js controlador (por ejemplo, mongosh) deben actualizarse para que utilicen Node.js la versión 6.13.1 o superior del controlador.

En el siguiente ejemplo, sustituya cada uno por la información de su *`user input placeholder`* clúster.

```
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

El código siguiente demuestra cómo conectarse a Amazon DocumentDB utilizando PHP cuando TLS está deshabilitado.

En el siguiente ejemplo, reemplaza cada una de *user input placeholder* ellas por la información de tu clúster.

```
<?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

El código siguiente demuestra cómo conectarse a Amazon DocumentDB utilizando Go cuando TLS está deshabilitado.

En el siguiente ejemplo, reemplaza cada una de *user input placeholder* ellas por la información de tu clúster.

```
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

El código siguiente demuestra cómo conectarse a Amazon DocumentDB utilizando Java cuando TLS está deshabilitado.

En el siguiente ejemplo, reemplaza cada una de *user input placeholder* ellas por la información de tu clúster.

```
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

El código siguiente demuestra cómo conectarse a Amazon DocumentDB utilizando C# / .NET cuando TLS está deshabilitado.

En el siguiente ejemplo, reemplaza cada una de *user input placeholder* ellas por la información de tu clúster.

```
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

El siguiente código demuestra cómo conectarse a Amazon DocumentDB y realizar consultas mediante la versión más reciente, mongosh, o la versión anterior del intérprete de comandos de mongo cuando TLS está deshabilitado.

Conexión con Amazon DocumentDB con mongosh

> **Important**
>
> Existe una limitación conocida en relación con Node.js los controladores anteriores a la versión 6.13.1, que actualmente no son compatibles con la autenticación de identidad de IAM en Amazon DocumentDB. Node.js Los controladores y las herramientas que utilizan el Node.js controlador (por ejemplo, mongosh) deben actualizarse para que utilicen Node.js la versión 6.13.1 o superior del controlador.

En los siguientes ejemplos, sustituya cada uno de ellos por la información *`user input placeholder`* de su clúster.

```
mongosh --host cluster-end-point:27017 --username sample-user --password password --
retryWrites false
```

Conexión con Amazon DocumentDB con la versión anterior del intérprete de comandos de mongo

Si usa IAM, debe usar una versión anterior del intérprete de comandos de mongo. Introduzca una de las siguientes opciones de comando:

```
mongo --host cluster-end-point:27017 --username sample-user --password password
```

Si utiliza una versión igual o superior a la 4.2, utilice el siguiente código para conectarse. Amazon DocumentDB no admite el reintento de las escrituras. Si utiliza el intérprete de comandos de mongo heredado (no mongosh), no incluya el comando `retryWrites=false` en ninguna cadena de código. El reintento de las escrituras está desactivado de forma predeterminada. Incluir `retryWrites=false` podría provocar un error en comandos de lectura normales.

```
mongo --host cluster-end-point:27017 --username sample-user --password password
```

Pruebe la conexión

1. Insertar un documento.

   ```
   db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
   ```

2. Busque el documento que ha insertado anteriormente.

   ```
   db.myTestCollection.find({'hello':'Amazon DocumentDB'})
   ```

R

En el siguiente código, se muestra cómo conectarse a Amazon DocumentDB con R usando mongolite (https://jeroen.github.io/mongolite/) cuando TLS está deshabilitado.

En el siguiente ejemplo, sustituya cada una de *user input placeholder* ellas por la información de su clúster.

```
#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

El código siguiente demuestra cómo conectarse a Amazon DocumentDB con Ruby cuando TLS está deshabilitado.

En el siguiente ejemplo, reemplaza cada una de *user input placeholder* ellas por la información de tu clúster.

```
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
```

Compresión de red en Amazon DocumentDB

Con la compresión mediante protocolos cableados, puede reducir los datos que la aplicación envía a través de la red. Amazon DocumentDB 8.0.1 y versiones posteriores admiten la compresión. Obtiene un rendimiento mejorado para las cargas de trabajo que transfieren documentos o conjuntos de resultados de gran tamaño. Después de habilitar la compresión, el controlador y el servidor intercambian los mensajes comprimidos.

Cuando se habilita la compresión de red, se intercambian los recursos de la CPU por un ancho de banda de red reducido. Habilite la compresión cuando la carga de trabajo esté vinculada a la red y transfiera grandes cargas útiles. En el caso de las solicitudes y respuestas pequeñas, es posible que no obtenga ningún beneficio neto, ya que la sobrecarga de compresión puede superar los ahorros.

Compresores compatibles

Amazon DocumentDB es compatible con el `snappy` compresor. Amazon DocumentDB no admite otros compresores que ofrecen los controladores MongoDB, como y. `zlib` `zstd`

Si un controlador solicita una lista de compresores, Amazon DocumentDB solo negocia los que son compatibles. Por ejemplo, si un controlador ofrece un `snappy` `yzstd`, Amazon DocumentDB lo selecciona y lo ignora. `snappy` `zstd` Si un controlador solo ofrece compresores no compatibles, Amazon DocumentDB no comprime la conexión. En este caso, el controlador y el servidor intercambian mensajes sin compresión.

Habilitar la compresión

Para habilitar la compresión, añada la `compressors` opción a la cadena de conexión y configúrela en`snappy`. En la siguiente cadena de conexión *`usernamepassword`*, sustituya y *`docdb-cluster-endpoint`* por sus propios valores:

```
mongodb://username:password@docdb-cluster-endpoint:27017/?
tls=true&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false&compressors=snappy
```

Cuando utilice un controlador MongoDB, defina la misma opción mediante las opciones de conexión del controlador. La mayoría de los controladores requieren una dependencia adicional del lado del cliente para realizar una compresión rápida:

- El Node.js controlador necesita el paquete. `snappy`
- El controlador Python (PyMongo) requiere el `python-snappy` paquete. En Linux, `python-snappy` también requiere la biblioteca snappy del sistema (por ejemplo, `snappy-devel` through `dnf` o`yum`).

Para obtener el nombre exacto de la opción y los paquetes necesarios, consulte la documentación del controlador.

> **Note**
>
> Si falta la dependencia de compresión del lado del cliente, el controlador se conecta correctamente. Sin embargo, intercambia mensajes sin compresión y no devuelve ningún error. Para confirmar que la compresión está activa, compruebe el compresor negociado tal y como se describe enVerifica la compresión.

El siguiente ejemplo de código de Python se conecta a su clúster con la compresión rápida habilitada y utiliza un `except` bloque `try` y para gestionar los errores de conexión:

```
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 compresión

Para confirmar que un controlador y Amazon DocumentDB negociaron la compresión, inspeccione el `compression` campo que devuelve el comando `hello` MongoDB wire protocol. Cuando el controlador y Amazon DocumentDB negocian la compresión, este campo muestra los compresores acordados.

Conexión a un clúster de Amazon DocumentDB desde fuera de una Amazon VPC

Los clústeres de Amazon DocumentDB (compatible con MongoDB) se implementan en una instancia de Amazon Virtual Private Cloud (Amazon VPC). Las instancias de Amazon EC2 u otros AWS servicios que estén implementados en la misma Amazon VPC pueden acceder a ellos directamente. Además, las instancias EC2 u otros AWS servicios de diferentes VPC de la misma Región de AWS región o de otras pueden acceder a Amazon DocumentDB mediante la interconexión de VPC.

Sin embargo, supongamos que su caso de uso requiere que usted o su aplicación tengan acceso a los recursos de Amazon DocumentDB desde fuera de la VPC del clúster. En ese caso, puede utilizar la tunelización SSH (denominada también reenvío de puertos) para obtener acceso a sus recursos de Amazon DocumentDB.

La descripción detallada de la tunelización SSH queda fuera del alcance de este tema. Si desea obtener más información sobre la tunelización SSH, consulte los siguientes temas:

- Túnel SSH

- SSH Port Forwarding Example, en concreto la sección Local Forwarding

Para crear un túnel SSH, necesita una instancia de Amazon EC2 que se ejecute en la misma VPC de Amazon que el clúster de Amazon DocumentDB. Puede usar una instancia EC2 existente en la misma VPC que el clúster o crear una. Para obtener más información, consulte el tema correspondiente a su sistema operativo:

- Introducción a las instancias Linux de Amazon EC2
- Introducción a las instancias de Windows de Amazon EC2

Es posible que normalmente se conecte a una instancia EC2 con el siguiente comando:

```
ssh -i "ec2Access.pem" ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com
```

Si es así, puede configurar un túnel de SSH en el clúster de Amazon DocumentDB `sample-cluster.node.us-east-1.docdb.amazonaws.com` ejecutando el siguiente comando en el equipo local. La marca `-L` se utiliza para el reenvío de un puerto local. Cuando utilice un túnel SSH, conéctese al clúster mediante el punto de conexión del clúster y no intente conectarse en el modo de conjunto de réplicas (es decir, especificándolo `replicaSet=rs0` en la cadena de conexión), ya que se generará un error.

```
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
```

Una vez creado el túnel SSH, cualquier comando que envíe a `localhost:27017` se reenvía al clúster de Amazon DocumentDB `sample-cluster` que se ejecuta en la VPC de Amazon. Si la seguridad de la capa de transporte (TLS) está habilitada en el clúster de Amazon DocumentDB, tiene que descargar la clave pública de Amazon DocumentDB desde https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem . La siguiente operación descarga este archivo:

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

> **Note**
>
> TLS está habilitado de forma predeterminada para los nuevos clústeres de Amazon DocumentDB. No obstante, sí puede deshabilitarlo. Para obtener más información, consulte Administración de la configuración de TLS del clúster de Amazon DocumentDB.

Para conectarse a su clúster de Amazon DocumentDB desde fuera de Amazon VPC, utilice el comando siguiente.

```
mongo --sslAllowInvalidHostnames --ssl --sslCAFile global-bundle.pem --username
 <yourUsername> --password <yourPassword>
```

Conexión a Amazon DocumentDB como conjunto de réplicas

Cuando desarrolle con Amazon DocumentDB, conéctese a su clúster como un conjunto de réplicas y distribuya las lecturas a las instancias de réplica mediante las capacidades de preferencias de lectura integradas de su controlador. En esta sección se profundiza en lo que esto significa y se describe cómo puede conectarse a su clúster de Amazon DocumentDB como un conjunto de réplicas utilizando Python como ejemplo.

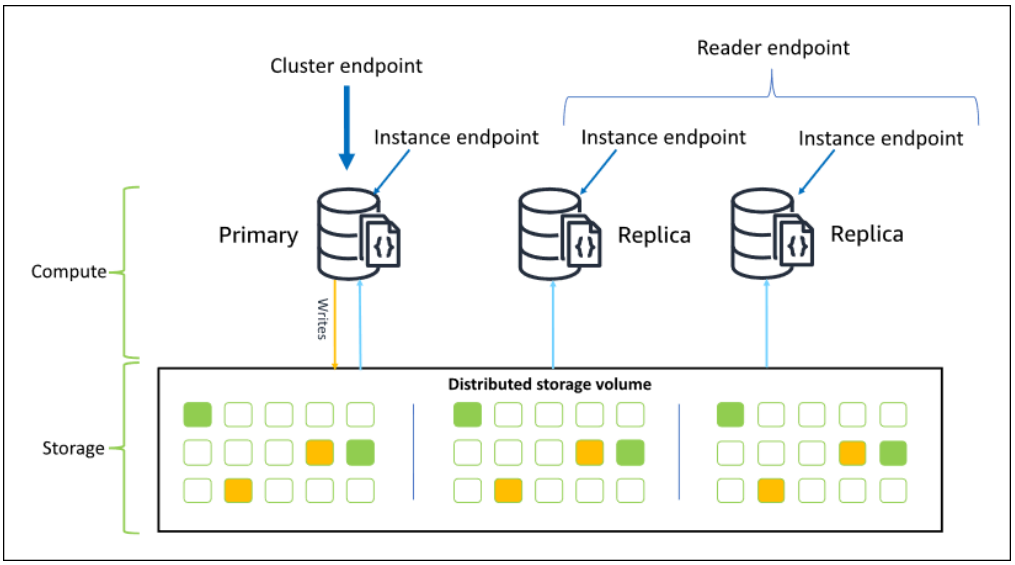
Amazon DocumentDB tiene tres puntos de conexión que puede utilizar para conectarse al clúster:

- Punto de conexión de clúster
- Punto de conexión del lector
- Puntos de conexión de instancia

Cuando se conecte a Amazon DocumentDB, utilice el punto final del clúster. Se trata de un CNAME que apunta a la instancia principal del clúster, tal y como se muestra en el siguiente diagrama.

Cuando utilice un túnel SSH, conéctese a su clúster mediante el punto de enlace del clúster. No te conectes en el modo de conjunto de réplicas (especifícalo `replicaSet=rs0` en la cadena de conexión), ya que se producirá un error.

> **Note**
>
> Para obtener más información acerca de los puntos de conexión de sitio web de Amazon DocumentDB, consulte Puntos de conexión de Amazon DocumentDB.



Con el punto de conexión de clúster, puede conectarse al clúster en modo de conjunto de réplicas. A continuación, puede utilizar las funciones integradas del controlador de preferencias de lectura. En el siguiente ejemplo, al especificar `/?replicaSet=rs0`, el SDK entiende que desea conectarse como conjunto de réplicas. Si omite `/?replicaSet=rs0'`, el cliente dirige todas las solicitudes al punto de conexión de clúster, es decir, la instancia principal.

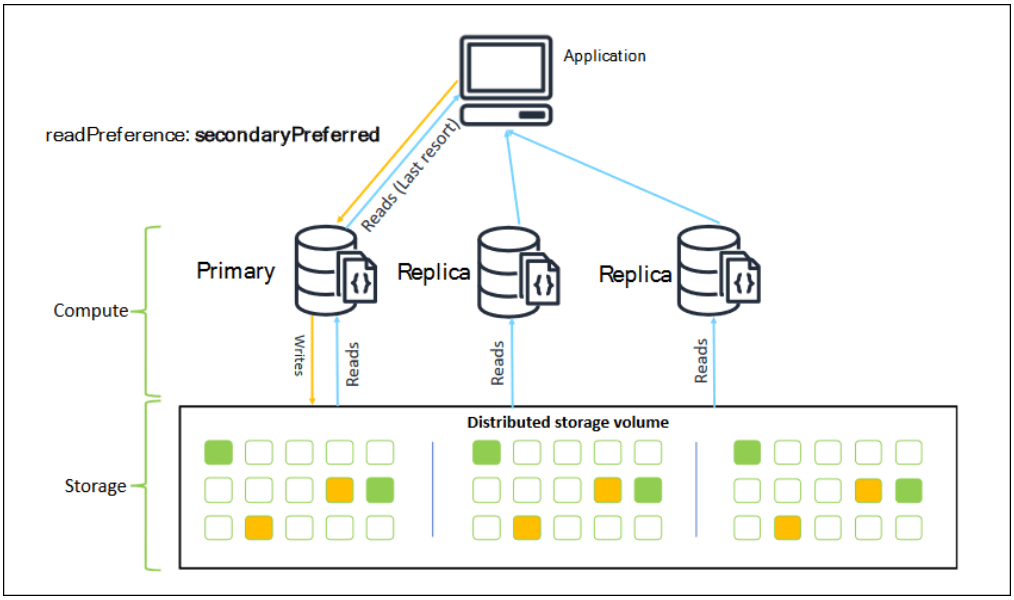
```
## Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ## replica set and specify the read preference as secondary preferred
   client = pymongo.MongoClient('mongodb://<user-name>:<password>@mycluster.node.us-
east-1.docdb.amazonaws.com:27017/?replicaSet=rs0')
```

La ventaja de conectarse como conjunto de réplicas es que permite al SDK detectar automáticamente la topografía del clúster, lo que incluye cuándo se añaden o se eliminan instancias del clúster. Posteriormente, puede utilizar el clúster de forma más eficiente dirigiendo las solicitudes de lectura a las instancias de réplica.

Cuando se conecta como conjunto de réplicas, puede especificar la `readPreference` para la conexión. Si especifica una preferencia de lectura de `secondaryPreferred`, el cliente dirige las consultas de lectura a las réplicas y las consultas de escritura, a la instancia principal (como

en el siguiente diagrama). Así, se aprovechan mejor los recursos del clúster. Para obtener más información, consulte Opciones de preferencia de lectura.

```
## 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')
```



Las lecturas de las réplicas de Amazon DocumentDB presentan consistencia final. Devuelven los datos en el mismo orden en que se escribieron en la instancia principal y, a menudo, hay un retardo de replicación inferior a 50 ms. Puede supervisar el retraso de las réplicas de su clúster utilizando las CloudWatch métricas de Amazon `DBInstanceReplicaLag` y`DBClusterReplicaLagMaximum`. Para obtener más información, consulte Supervisión de Amazon DocumentDB con CloudWatch.

A diferencia de la arquitectura monolítica tradicional de las bases de datos, Amazon DocumentDB separa el almacenamiento y la computación. Dada esta arquitectura moderna, le animamos a escalar la lectura en las instancias de réplica. Las lecturas en las instancias de réplica no bloquean la replicación de las escrituras desde la instancia principal. Puede añadir hasta 15 instancias de réplica de lectura en un clúster y escalar horizontalmente a millones de lecturas por segundo.

La ventaja principal de conectarse como conjunto de réplicas y distribuir las lecturas a las réplicas es que aumenta los recursos generales del clúster disponibles para utilizar la aplicación. Como práctica recomendada, conéctese como un conjunto de réplicas. Esto es especialmente beneficioso en los siguientes escenarios:

- Si utiliza casi el 100 % de la CPU en la instancia principal.
- Si la proporción de aciertos de la caché del búfer es casi cero.
- Si alcanza los límites de conexión o cursor de una instancia.

Escalar el tamaño de una instancia de clúster es una opción y, en algunos casos, puede ser la mejor manera de escalar el clúster. Sin embargo, también deberá plantearse cómo aprovechar mejor las réplicas que ya tiene en el clúster. Esto le permite aumentar la escala sin el mayor coste de utilizar un tipo de instancia mayor. También le recomendamos que supervise estos límites y avise al respecto (es decir `CPUUtilization`, `DatabaseConnections`, y`BufferCacheHitRatio`) que utilice CloudWatch alarmas para saber si un recurso se está consumiendo en exceso.

Para obtener más información, consulte los temas siguientes:

- Prácticas recomendadas para Amazon DocumentDB
- Cuotas de Amazon DocumentDB

Uso de las conexiones del clúster

Plantéese la posibilidad de utilizar todas las conexiones del clúster. Por ejemplo, una instancia `r5.2xlarge` tiene un límite de 4500 conexiones (y 450 cursores abiertos). Si crea un clúster de Amazon DocumentDB de tres instancias y se conecta únicamente a la instancia principal mediante el punto de conexión de clúster, los límites de conexiones y cursores abiertos del clúster son de 4500 y 450 respectivamente. Podría alcanzar estos límites al crear aplicaciones que utilizan muchos procesos de trabajo que se ponen en marcha en contenedores. Los contenedores abren una serie de conexiones a la vez y saturan el clúster.

En su lugar, podría conectarse al clúster de Amazon DocumentDB como conjunto de réplicas y distribuir las lecturas a las instancias de réplica. A continuación, podría triplicar de forma eficaz el número de conexiones y cursores disponibles en el clúster a 13 500 y 1350 respectivamente. Añadir más instancias al clúster solo aumenta el número de conexiones y cursores para las cargas de trabajo de lectura. Si necesita aumentar el número de conexiones para las escrituras en el clúster, recomendamos que aumente el tamaño de la instancia.

> **Note**
>
> El número de conexiones para las instancias `large`, `xlarge` y `2xlarge` aumenta con el tamaño de instancia hasta 4500. El número máximo de conexiones por instancia para

instancias `4xlarge` o superiores es 4500. Para obtener más información sobre los límites por tipos de instancia, consulte Cuotas de instancias.

Normalmente, no recomendamos que se conecte al clúster con la preferencia de lectura de `secondary`. Esto se debe a que, si no hay instancias de réplica en el clúster, las lecturas generarán un error. Por ejemplo, suponga que tiene un clúster de Amazon DocumentDB de dos instancias con una instancia principal y una réplica. Si la réplica tiene un problema, se generará un error en las solicitudes de lectura de un grupo de conexiones establecido como `secondary`. La ventaja de `secondaryPreferred` es que, si el cliente no encuentra una instancia de réplica adecuada a la que conectarse, recurre a la instancia principal para las lecturas.

Varios grupos de conexiones

En algunos casos, las lecturas de una aplicación deben contar con consistencia de lectura tras escritura, que solo se puede distribuir desde la instancia principal en Amazon DocumentDB. En estos casos, podría crear dos grupos de conexiones del cliente: uno para escrituras y otro para lecturas que necesiten consistencia de lectura tras escritura. Para ello, el código tendría un aspecto similar al siguiente.

```
## 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')
```

Otra opción consiste en crear un único grupo de conexiones y sobrescribir la preferencia de lectura en una colección determinada.

```
##Specify the collection and set the read preference level for that collection
col = db.review.with_options(read_preference=ReadPreference.SECONDARY_PREFERRED)
```

Resumen

Para utilizar mejor los recursos del clúster, conéctese mediante el modo de conjunto de réplicas. Si es adecuado para la aplicación, puede escalar la lectura de la aplicación distribuyendo las lecturas a las instancias de réplica.

Conexión a un clúster de Amazon DocumentDB desde Studio 3T

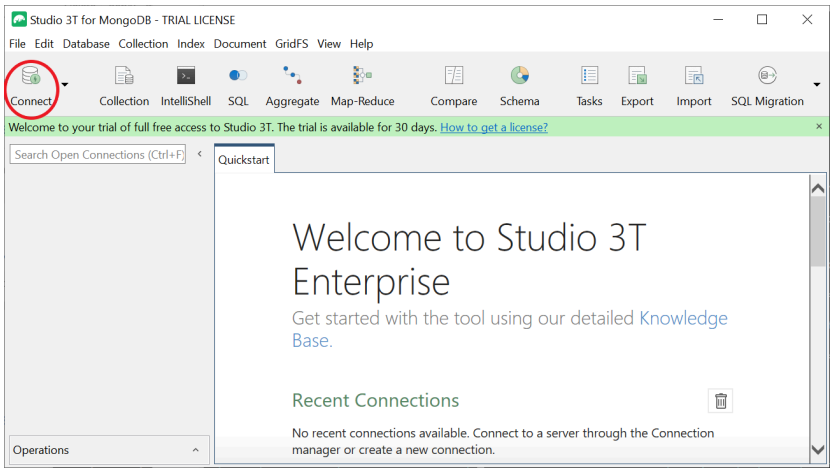
Studio 3T es una GUI e IDE popular para desarrolladores e ingenieros de datos que trabajan con MongoDB. Ofrece varias funciones potentes: vistas de sus datos en árbol, tabla y JSON, sencillas import/export en CSV, JSON, SQL y una opción de consulta flexible BSON/mongodump, una interfaz de usuario visual de arrastrar y soltar, una consola mongo integrada con finalización automática, un editor de canalizaciones de agregación y soporte para consultas SQL.

Requisitos previos

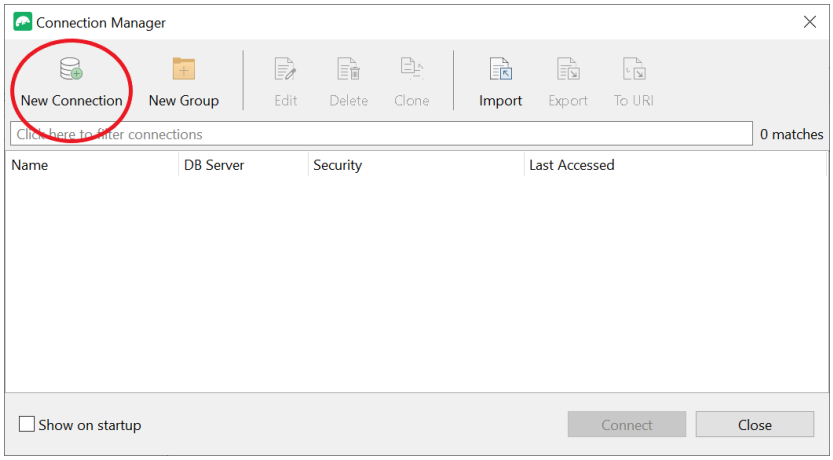
- Si aún no tiene un clúster de Amazon DocumentDB que utilice Amazon EC2 como bastion/jump host, consulte. Conectarse mediante Amazon EC2
- Si no tiene Studio 3T, descárguelo e instálelo.

Conexión con Studio 3T

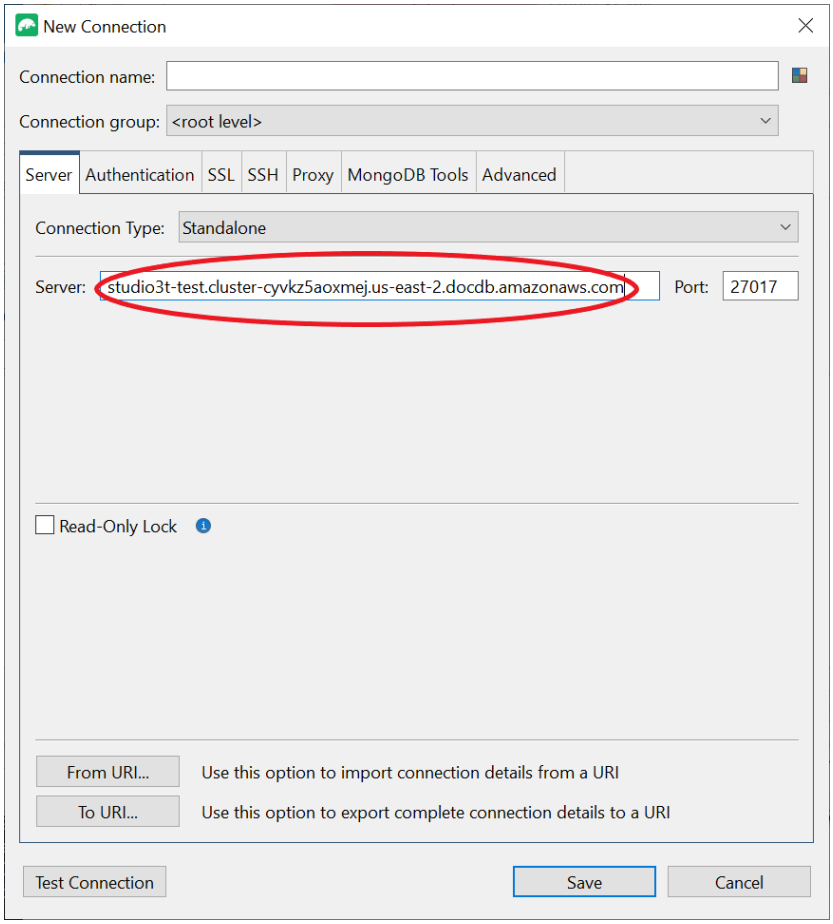
1. Seleccione Conectar en la esquina superior izquierda de la barra de herramientas.



2. Seleccione Nueva conexión en la esquina superior izquierda de la barra de herramientas.

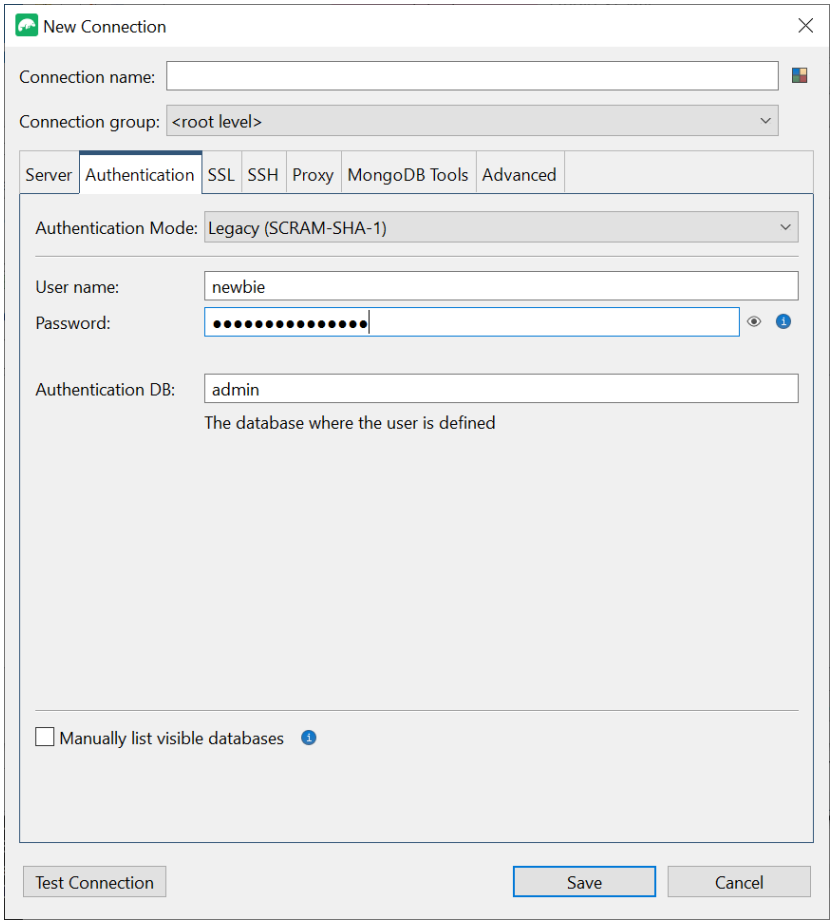


3. En la pestaña Servidor, en el campo Servidor, introduzca la información del punto de conexión del clúster.

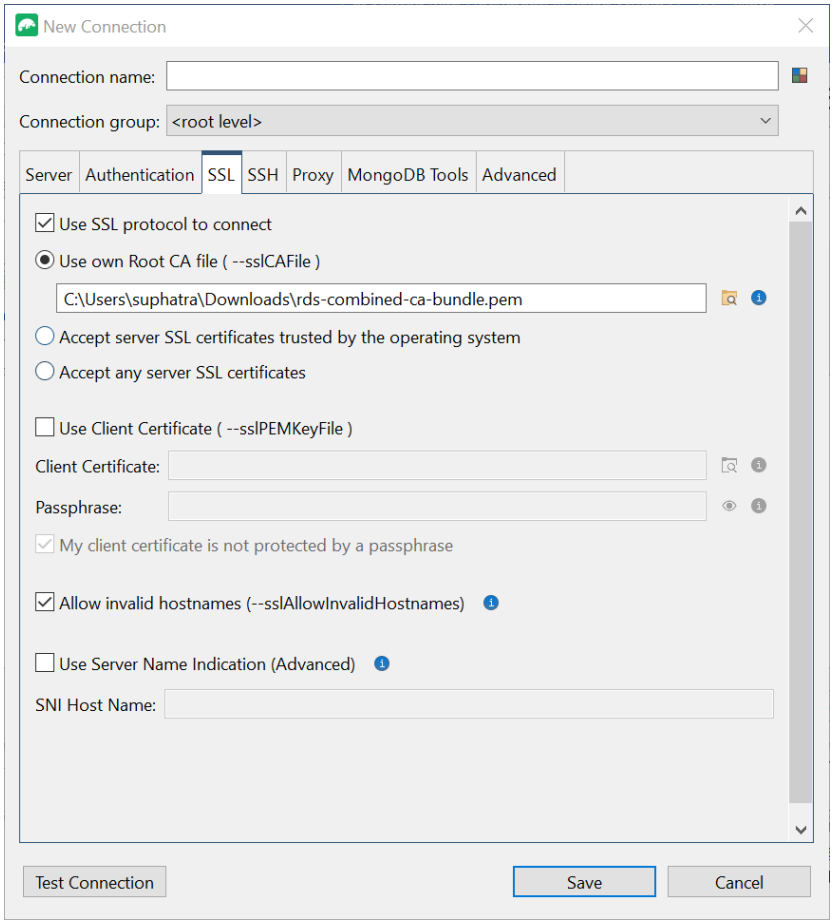


> **Note**
>
> Para encontrar el punto final de su clúster, consulte. Búsqueda del punto de conexión de una instancia

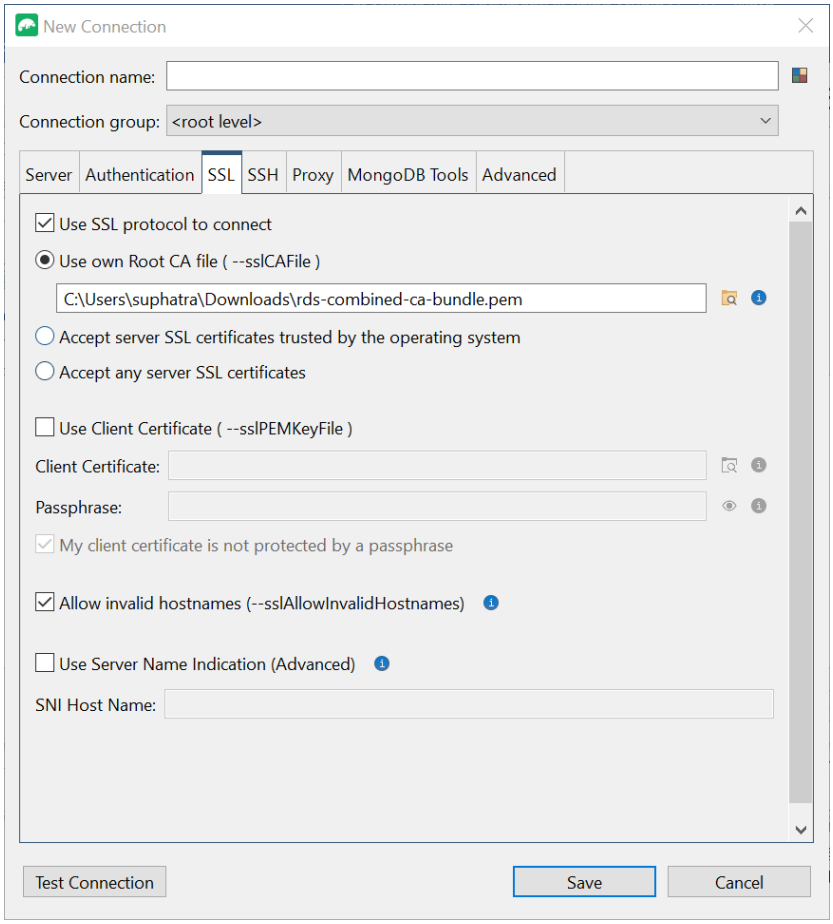
4. Elija la pestaña Autenticación y seleccione Heredado en el menú desplegable Modo de autenticación.



5. Introduzca su nombre de usuario y sus credenciales en los campos Nombre de usuario y Contraseña.
6. Seleccione la pestaña SSL y marque la casilla Usar el protocolo SSL para conectarse.



7. Seleccione Usar un archivo CA raíz propio. A continuación, añada el certificado de Amazon DocumentDB (puede omitir este paso si SSL está deshabilitado en su clúster de DocumentDB). Marque la casilla para permitir nombres de host no válidos.



> **Note**
>
> ¿No tiene el certificado? Puede descargarlo con el siguiente comando:
> `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

8. Si se conecta desde una máquina cliente externa a Amazon VPC, debe crear un túnel SSH. Hará esto en la pestaña SSH.

   a. Marque la casilla Usar túnel SSH e introduzca la dirección SSH en el campo Dirección SSH. Este es el DNS público (IPV4) de su instancia. Puede obtener esta URL desde su Consola de administración de Amazon EC2.

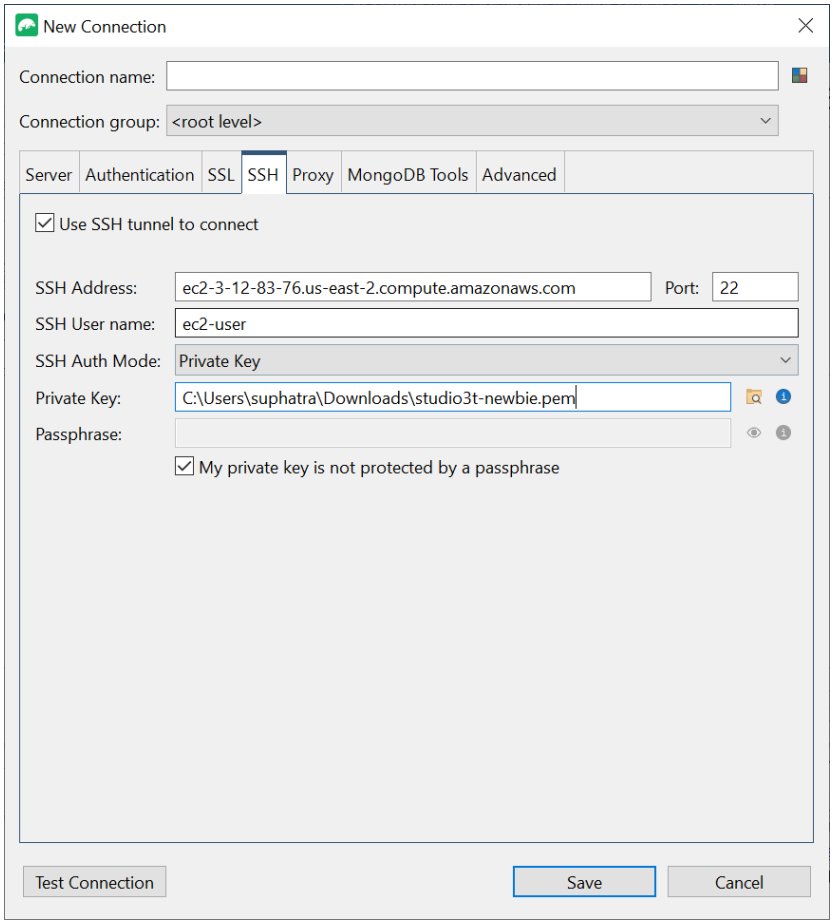
   b. Introduzca su nombre de usuario. Este es el nombre de usuario de su instancia de Amazon EC2.

   c. Como Modo de autenticación SSH, seleccione Clave privada. En el campo Clave privada, elija el icono del buscador de archivos para localizar y elegir la clave privada de su instancia

de Amazon EC2. Este es el archivo .pem (par de claves) que guardó al crear la instancia en la consola de Amazon EC2.

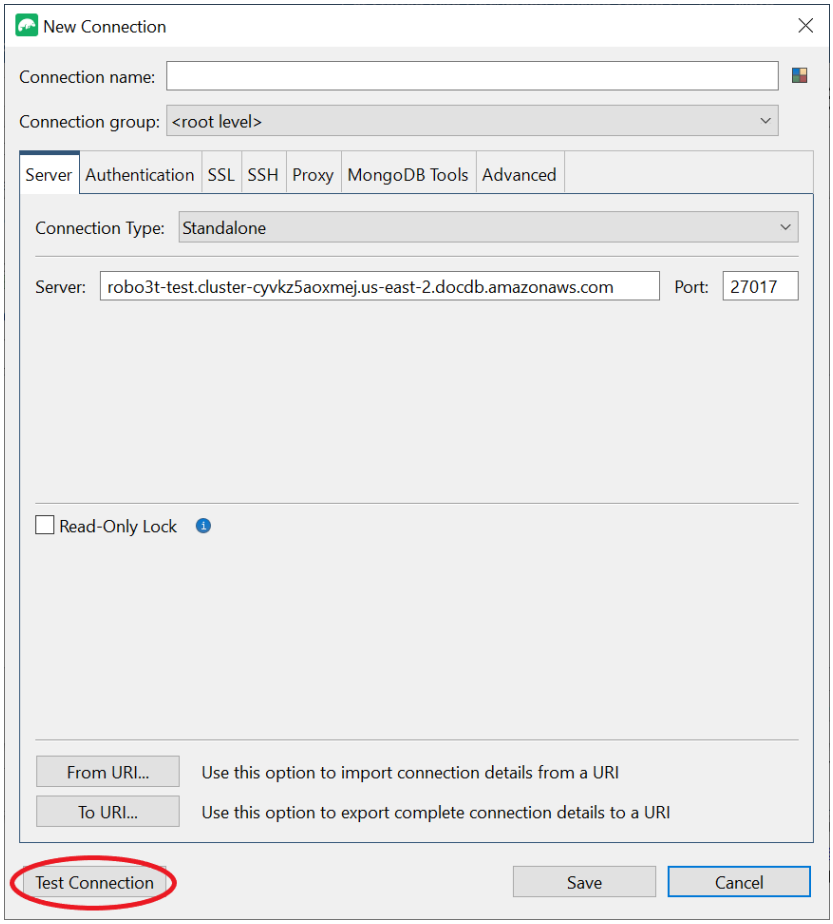
d. Si está en un equipo Linux/macOS cliente, es posible que tenga que cambiar los permisos de la clave privada mediante el siguiente comando:

```
chmod 400 /fullPathToYourPemFile/<yourKey>.pem
```

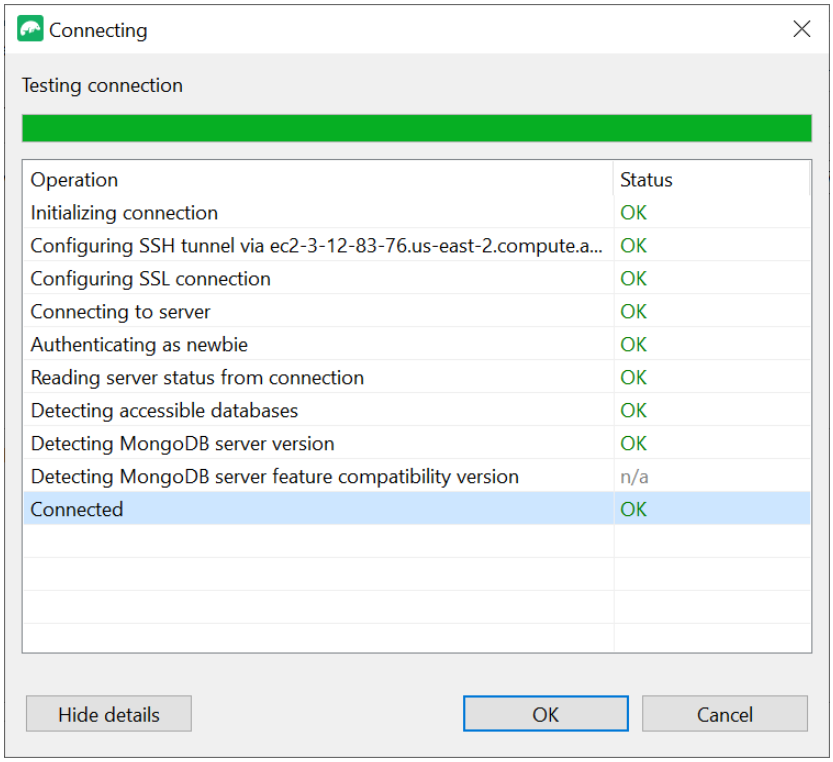


> **Note**
>
> Esta instancia de Amazon EC2 debe estar en la misma VPC de Amazon y grupo de seguridad que el clúster de DocumentDB. Puede obtener la dirección SSH, el nombre de usuario y la clave privada en su Consola de administración de Amazon EC2.

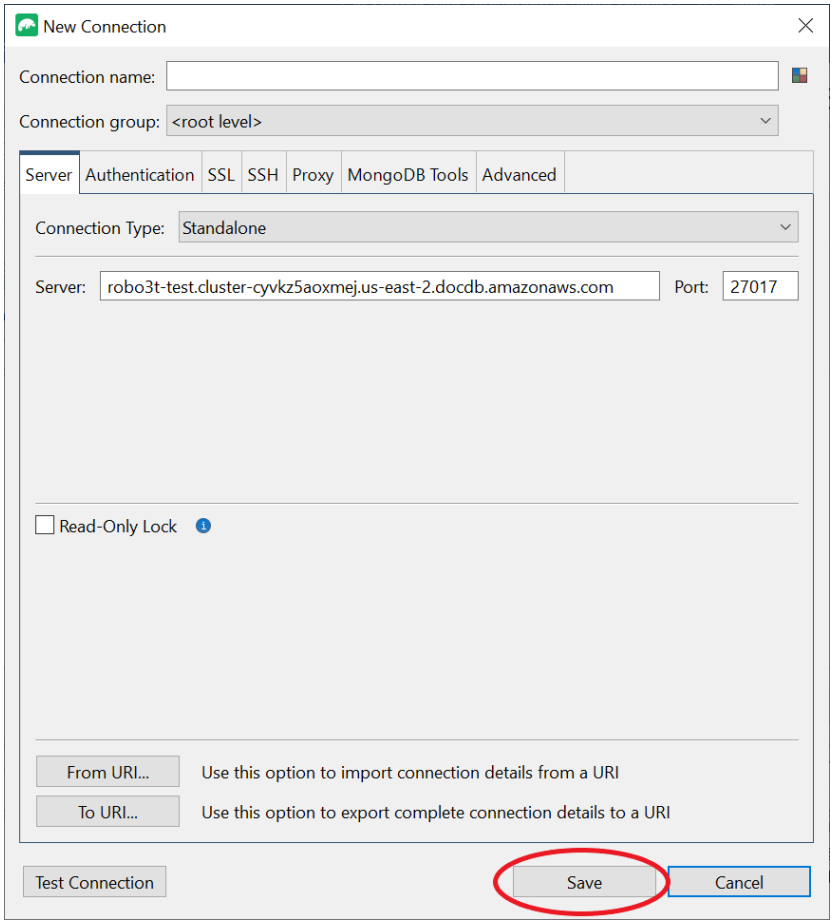
9. Ahora pruebe la configuración pulsando el botón Probar conexión.



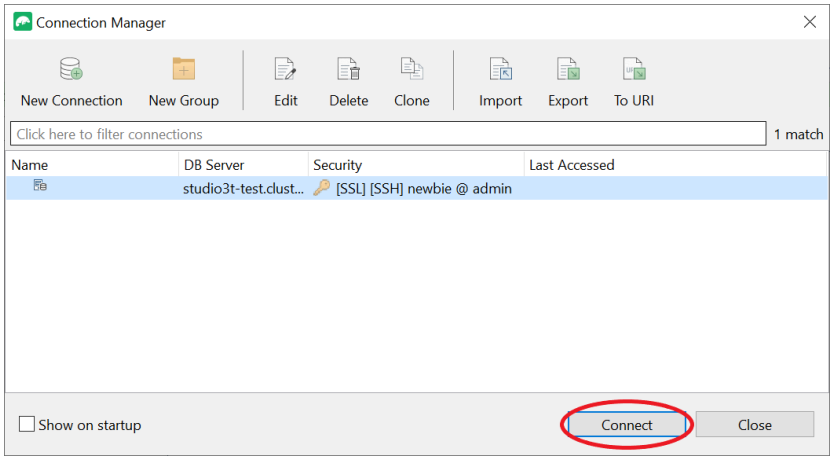
10. Una ventana de diagnóstico debería cargar una barra verde para indicar que la prueba se ha realizado correctamente. Ahora pulse Aceptar para cerrar la ventana de diagnóstico.



11. Elija Guardar para guardar la conexión y usarla en el futuro.



12. Ahora seleccione su clúster y elija Conectar.



¡Enhorabuena! Se ha conectado correctamente a su clúster de Amazon DocumentDB a través de Studio 3T.

Connect a Amazon DocumentDB mediante DataGrip

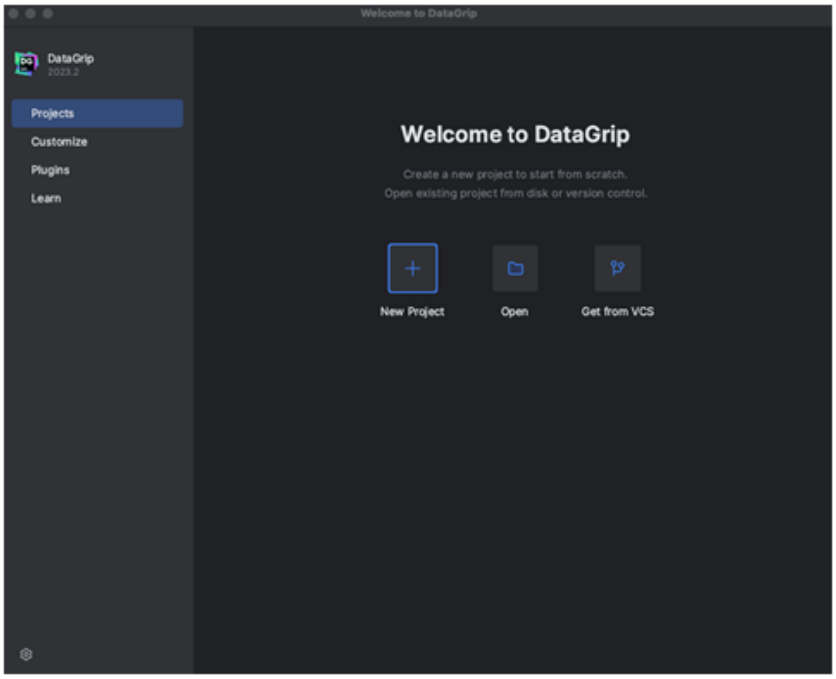
DataGripes un potente entorno de desarrollo integrado (IDE) que admite varios sistemas de bases de datos, incluido Amazon DocumentDB. En esta sección, se explican los pasos para conectarse a su clúster de Amazon DocumentDB mediante una interfaz gráfica DataGrip, lo que le permite administrar y consultar sus datos fácilmente.

Requisitos previos

- DataGrip IDE instalado en su máquina. Puede descargarlo desde JetBrains.
- Una instancia de Amazon EC2 que se ejecute en la misma VPC que el clúster de Amazon DocumentDB. Utilizará esta instancia para establecer un túnel seguro desde su máquina local hasta el clúster de Amazon DocumentDBCluster. Siga estas instrucciones sobre cómo Conectarse mediante Amazon EC2:
- Como alternativa a una instancia Amazon EC2, a una conexión VPN o si ya está accediendo a su AWS infraestructura mediante una VPN segura. Si prefiere esta opción, siga las instrucciones para acceder de forma segura a Amazon DocumentDB mediante. AWS Client VPN

Conéctese usando DataGrip

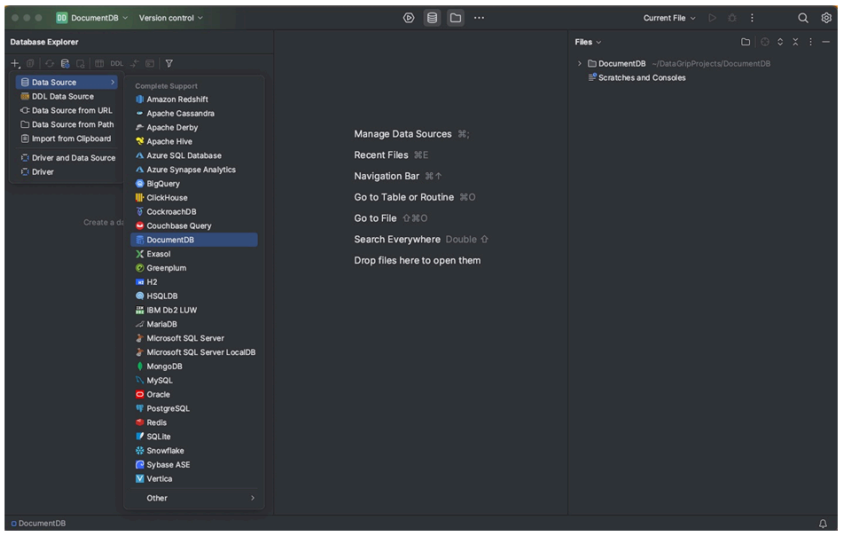
1. Inicie DataGrip en su computadora y cree un nuevo proyecto.



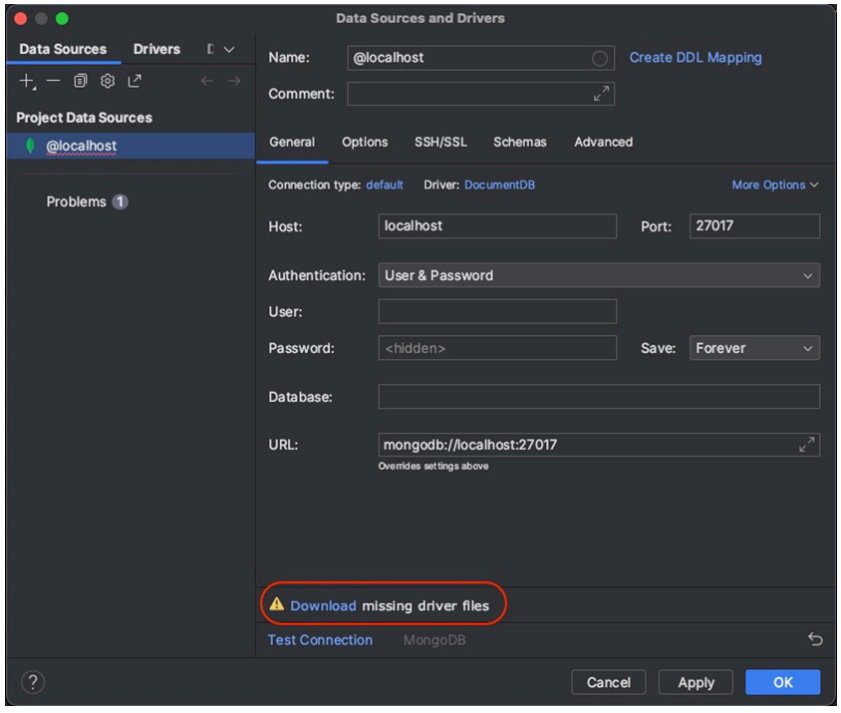
2. Agregue un nuevo origen de datos de una de las siguientes formas:

a. En el menú principal, vaya a Archivo — Nuevo — Origen de datos y seleccione DocumentDB

b. En el explorador de bases de datos, elija el nuevo icono (+) de la barra de herramientas. Vaya al origen de datos y seleccione DocumentDB.



3. En la página Orígenes de datos de la pestaña General, compruebe si hay un enlace para descargar los archivos del controlador que faltan en la parte inferior del área de configuración de la conexión. Seleccione este enlace para descargar los controladores necesarios para interactuar con una base de datos. Para obtener un enlace de descarga directa, consulte los controladores JetBrains JDBC.



4. En la pestaña General, especifique los detalles de la conexión:

a. En el campo Host, especifique el punto de conexión del clúster de Amazon DocumentDB.

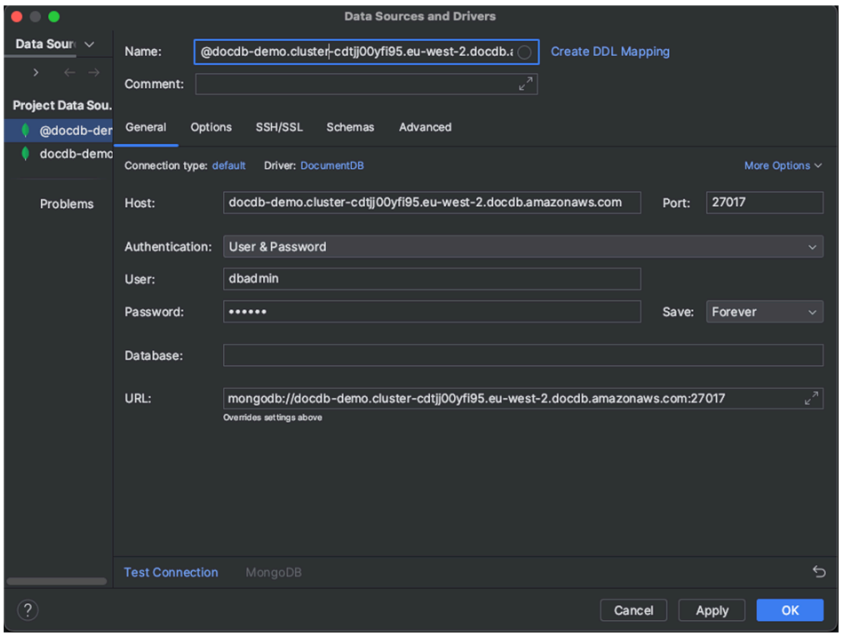
b. El puerto ya está establecido en 27017. Cámbielo si el clúster se implementó en un puerto diferente.

c. En Autenticación, elija Nombre de usuario y contraseña.

d. Introduzca la información de nombre de usuario y contraseña.

e. El campo Base de datos es opcional. Puede especificar la base de datos a la que desea conectarse.

f. El campo URL se completa automáticamente a medida que agrega los detalles anteriores.



5. En la SSH/SSLpestaña, habilite Usar el túnel SSH y, a continuación, elija el icono para abrir el cuadro de diálogo de configuración de SSH. Introduzca la información siguiente:

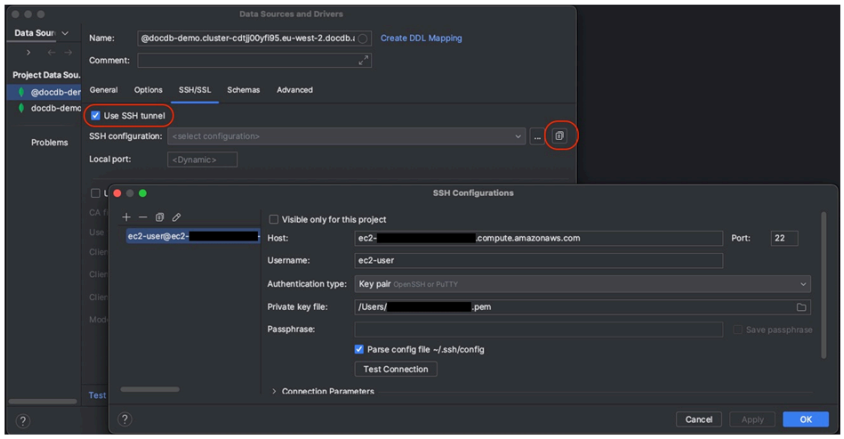
a. En el campo Host, introduzca el nombre de host de su instancia de Amazon EC2.

b. Introduzca el nombre de usuario y la contraseña de la instancia de Amazon EC2.

c. En Tipo de autenticación, elija Par de claves.

d. Introduzca su Archivo de clave privada.

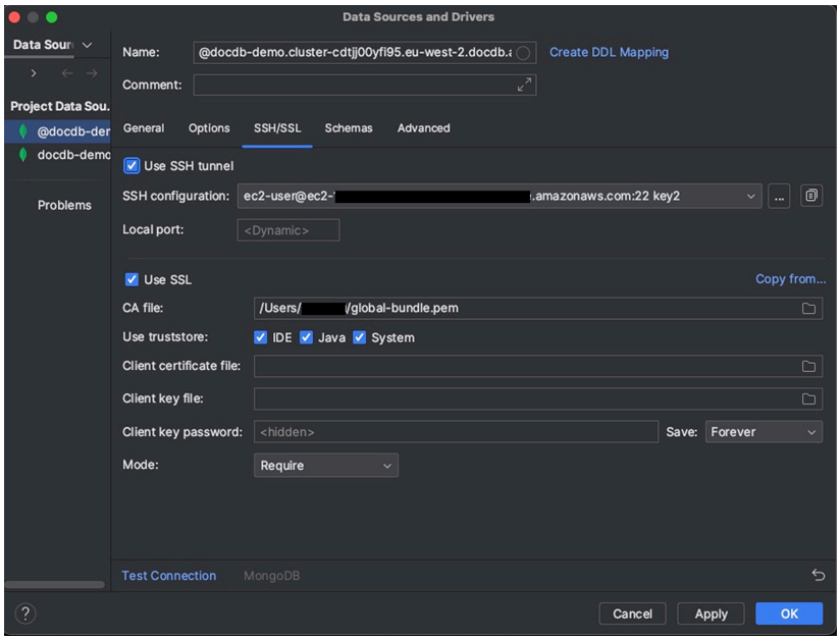
> **Note**
>
> Si utiliza la opción VPN, no es necesario configurar el túnel SSH.



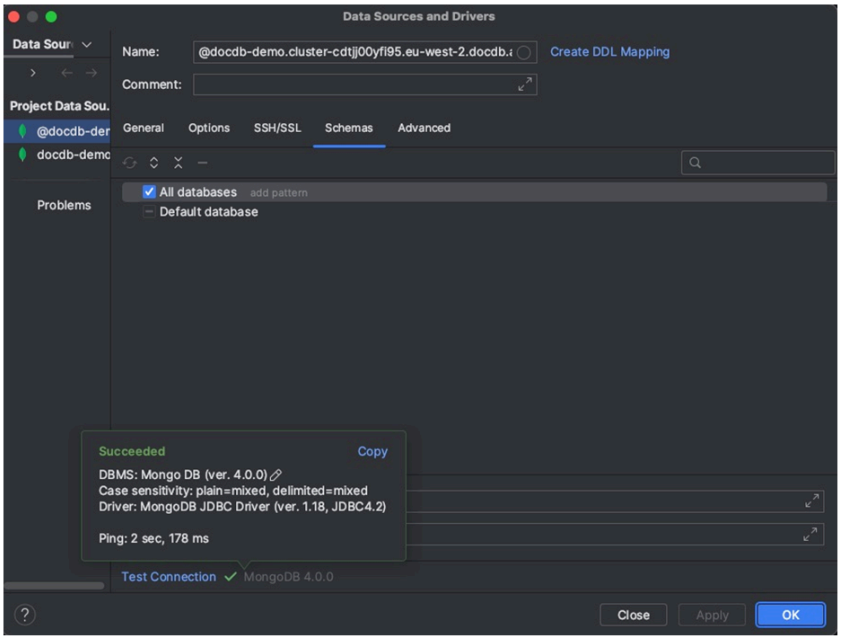
6. En la SSH/SSLpestaña, habilita Usar SSL. En el campo Archivo CA, introduzca la ubicación del archivo `global-bundle.pem` en su ordenador. En Modo, deje la opción Requerir.

> **Note**
>
> Puede descargar el certificado desde esta ubicación o con este comando: wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem

> **Note**
>
> Si se está conectando al clúster elástico de Amazon DocumentDB, no tiene que especificar el archivo CA. Deje marcada la opción Usar SSL y todas las demás opciones con sus valores predeterminados.



7. En la pestaña Esquemas, elija Todas las bases de datos o introduzca el filtro “*:*” en el campo del patrón del esquema. Seleccione Probar conexión para probar la conexión.



8. Una vez que la conexión se haya probado correctamente, pulse Aceptar para guardar la configuración de la fuente de datos.

DataGrip features

DataGrip proporciona varias funciones que le ayudan a trabajar con Amazon DocumentDB de forma eficaz:

- Editor SQL: escriba y ejecute SQL-like consultas en sus colecciones de DocumentDB mediante el editor SQL de. DataGrip
- Generador visual de consultas: utilice el generador visual de consultas para crear consultas de forma gráfica sin escribir el código SQL.
- Administración de esquemas: administre fácilmente el esquema de su base de datos, incluida la creación, modificación y eliminación de colecciones.
- Visualización de datos: vea y analice sus datos mediante varias herramientas de visualización disponibles en DataGrip.
- Exportación e importación de datos: transfiera datos entre Amazon DocumentDB y otras bases de datos mediante las funciones DataGrip de exportación e importación.

Consulte la DataGrip documentación oficial para obtener características y consejos más avanzados sobre cómo trabajar con Amazon DocumentDB y otros sistemas de bases de datos.

Conectarse mediante Amazon EC2

En esta sección se describe cómo configurar la conectividad entre un clúster de Amazon DocumentDB y Amazon EC2 y cómo acceder al clúster de Amazon DocumentDB desde la instancia de Amazon EC2.

Hay dos opciones para configurar la conexión de EC2:

- Conexión automática de la instancia de EC2 a una base de datos de Amazon DocumentDB: utilice la característica de conexión automática de la consola de EC2 para configurar automáticamente la conexión entre la instancia de EC2 y una base de datos de Amazon DocumentDB nueva o existente. Esta conexión permite que el tráfico se desplace entre la instancia de EC2 y la base de datos de Amazon DocumentDB. Esta opción se utiliza normalmente para probar y crear nuevos grupos de seguridad.
- Conexión manual de la instancia de EC2 a la base de datos de Amazon DocumentDB: configure la conexión entre la instancia de EC2 y la base de datos de Amazon DocumentDB mediante la configuración y la asignación manuales de los grupos de seguridad para que reproduzcan la

configuración que se crea mediante la característica de conexión automática. Esta opción se suele utilizar para cambiar configuraciones más avanzadas y usar los grupos de seguridad existentes.

Requisitos previos

Independientemente de la opción y antes de crear el primer clúster de Amazon DocumentDB, debe hacer lo siguiente:

Creación de una cuenta de Amazon Web Services (AWS)

Para empezar a utilizar Amazon DocumentDB, debe tener una cuenta de Amazon Web Services (AWS). La AWS cuenta es gratuita. Solo se paga por los servicios y los recursos que se utilicen.

Si no tiene una Cuenta de AWS, complete los siguientes pasos para crear una.

Para inscribirse en un Cuenta de AWS

1. https://signin.aws.amazon.com/signup¿Abierto? request_type=registrarse.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica o mensaje de texto e indicar un código de verificación en el teclado del teléfono.

   Cuando te registras en un, se crea un Cuenta de AWS. Usuario raíz de la cuenta de AWS El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar Tareas que requieren acceso de usuario raíz.

Si lo desea, configure los permisos necesarios AWS Identity and Access Management (IAM).

El acceso para administrar los recursos de Amazon DocumentDB, como los clústeres, las instancias y los grupos de parámetros de clústeres, requiere credenciales que AWS se puedan utilizar para autenticar las solicitudes. Para obtener más información, consulte Identity and Access Management para Amazon DocumentDB.

1. En la barra de búsqueda de Consola de administración de AWS, introduzca **`IAM`** y seleccione IAM en el menú desplegable que aparece.
2. Cuando esté en la consola de IAM, seleccione Usuarios en el panel de navegación.
3. Seleccione su nombre de usuario.

4. Elija Añadir permisos.
5. Seleccione Asociar directamente las políticas existentes.
6. Escriba `AmazonDocDBFullAccess` en la barra de búsqueda y selecciónelo en cuanto aparezca en los resultados de búsqueda.
7. Elija Next: Review (Siguiente: Revisión).
8. Elija Add permissions (Agregar permisos).

Creación de una Amazon Virtual Private Cloud (Amazon VPC)

Según el lugar en el que se Región de AWS encuentre, es posible que ya haya creado o no una VPC predeterminada. Si no cuenta con una VPC determinada, complete el paso 1 de la Introducción a Amazon VPC en la Guía del usuario de Amazon VPC. Esto tardará menos de cinco minutos.

Conexión automática de Amazon EC2

Temas

- Conexión automática de una instancia de EC2 a una nueva base de datos de Amazon DocumentDB
- Conexión automática de una instancia de EC2 a una base de datos de Amazon DocumentDB existente
- Descripción general de la conectividad automática con una instancia de EC2
- Visualización de los recursos de computación conectados

Antes de configurar una conexión entre una instancia de EC2 y una nueva base de datos de Amazon DocumentDB, asegúrese de cumplir con los requisitos descritos en Descripción general de la conectividad automática con una instancia de EC2. Si hace cambios en los grupos de seguridad después de configurar la conectividad, los cambios pueden afectar a la conexión entre la instancia de EC2 y la base de datos de Amazon DocumentDB.

> **Note**
>
> Solo puede configurar automáticamente una conexión entre una instancia de EC2 y una base de datos de Amazon DocumentDB automáticamente mediante la Consola de administración

de AWS. No puede configurar una conexión automáticamente con la AWS CLI API de Amazon DocumentDB.

Conexión automática de una instancia de EC2 a una nueva base de datos de Amazon DocumentDB

En el siguiente proceso se presupone que ha completado los pasos que se detallan en el tema Requisitos previos.

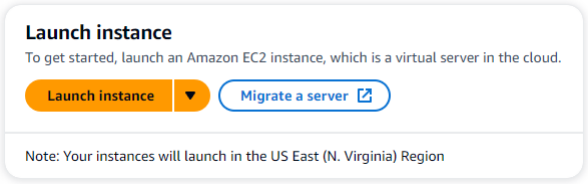
Steps

- Paso 1: creación de una instancia de Amazon EC2
- Paso 2: creación de un clúster de Amazon DocumentDB
- Paso 3: conexión a la instancia de Amazon EC2
- Paso 4: instalación del intérprete de comandos de MongoDB
- Paso 5: administración de TLS de Amazon DocumentDB
- Paso 6: conexión al clúster de Amazon DocumentDB
- Paso 7: inserción y consulta de datos
- Paso 8: exploración

Paso 1: creación de una instancia de Amazon EC2

En este paso, creará una instancia de Amazon EC2 en la misma región y Amazon VPC que utilizará más adelante para aprovisionar el clúster de Amazon DocumentDB.

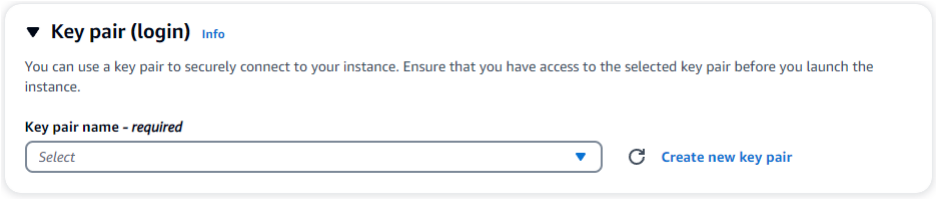
1. En el panel de la consola de Amazon EC2, seleccione Lanzar instancia.



2. Ingrese un nombre o identificador en el campo Nombre ubicado en la sección Nombres y etiquetas.
3. En la lista desplegable Imagen de máquina de Amazon (AMI), busque la AMI de Amazon Linux 2 y selecciónela.

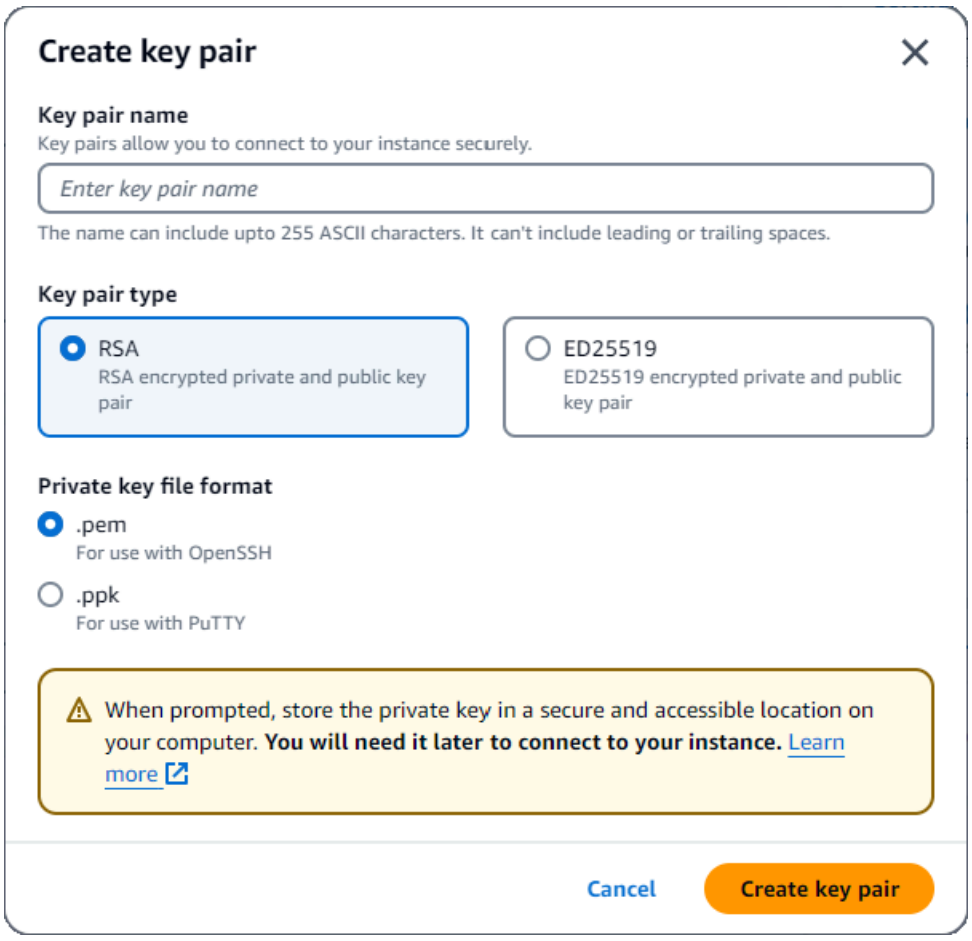


4. Busque y seleccione t3.micro en la lista desplegable de Tipo de instancia.
5. En la sección Par de claves (inicio de sesión), ingrese el identificador de un par de claves existente o elija Crear un par de claves nuevo.



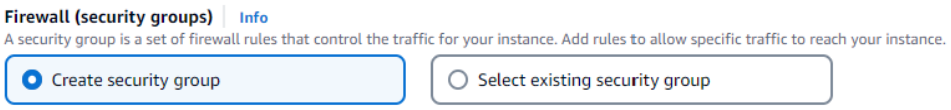
También debe proporcionar un par de claves de Amazon EC2.

- Si tiene un par de claves de Amazon EC2:
  1. Seleccione un par de claves, elija su par de claves de la lista.
  2. Debe tener el archivo de clave privada (archivo .pem o .ppk) a su disposición para poder iniciar sesión en la instancia de Amazon EC2.
- Si no tiene un par de claves de Amazon EC2:
  1. Seleccione Crear un par de claves nuevo y aparecerá el cuadro de diálogo Crear par de claves.
  2. Escriba un nombre en el campo Nombre del par de claves.
  3. Seleccione el Tipo de par de claves y el Formato de archivo de clave privada.
  4. Elija Crear par de claves.



> **Note**
>
> Por motivos de seguridad, le recomendamos encarecidamente que utilice un par de claves para la conectividad a Internet y SSH con la instancia de EC2.

6. Opcional: en la sección Configuración de red, en Firewall (grupos de seguridad), seleccione Crear grupo de seguridad.



Elija Crear grupo de seguridad (compruebe todas las reglas de tráfico que se aplican a su conectividad con EC2).

> **Note**
>
> Si desea utilizar un grupo de seguridad existente, siga las instrucciones de Conexión manual de Amazon EC2.

7. En la sección Resumen, revise la configuración de EC2 y, si es correcta, elija Lanzar instancia.

Paso 2: creación de un clúster de Amazon DocumentDB

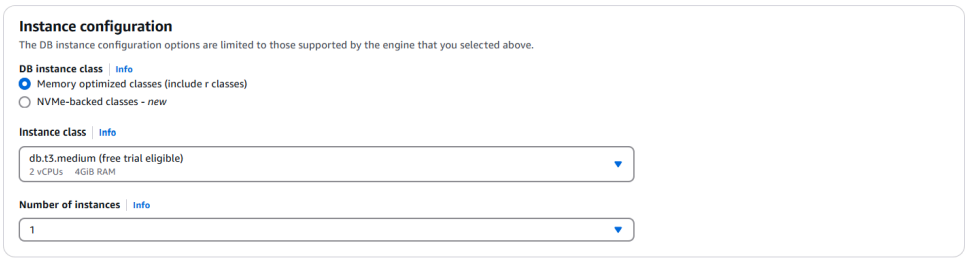
Mientras se aprovisiona la instancia de Amazon EC2, cree su clúster de Amazon DocumentDB.

1. Navegue hasta la consola de Amazon DocumentDB y elija Clústeres en el panel de navegación.
2. Seleccione Crear.
3. Deje la configuración de Tipo de clúster como predeterminada en Clúster basado en instancias.
4. En Configuración del clúster, ingrese un nombre único en Identificador de clúster. Tenga en cuenta que la consola cambiará todos los nombres de los clústeres a minúsculas independientemente de cómo se ingresen.

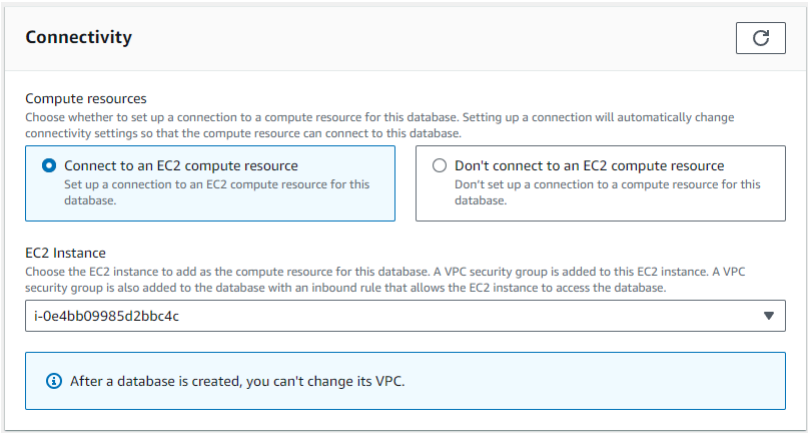
   Deje Versión del motor con su valor predeterminado de 5.0.0.
5. En Configuración del almacenamiento en clúster, deje la configuración predeterminada de Amazon DocumentDB estándar.
6. En Configuración de instancias:

   - En Clase de instancia de base de datos, elija Clases optimizadas para memoria (incluidas las clases r), que es la opción predeterminada.

     La otra opción de instancia son NVMe-backed las clases. Para obtener más información, consulte NVMe-backed instancias.
   - En Clase de instancia, elija el tipo de instancia que se adapte a sus necesidades. Para obtener una explicación más detallada de las clases de instancia, consulte Especificaciones de clases de instancias.
   - En Número de instancias, elija el número que mejor refleje sus necesidades. Recuerde que cuanto menor sea el número, menor será el costo y menor será el read/write volumen que puede administrar el clúster.



7. En Conectividad, elija Conectarse a un recurso de computación de EC2. Esta es la instancia de EC2 que creó en el paso 1.



> **Note**
>
> La conexión a un recurso de computación de EC2 crea automáticamente un grupo de seguridad para la conexión del recurso de computación de EC2 al clúster de Amazon DocumentDB. Cuando haya terminado de crear el clúster y desee ver el grupo de seguridad recién creado, navegue hasta la lista de clústeres y elija el identificador del clúster. En la pestaña Conectividad y seguridad, vaya a Grupos de seguridad y busque su grupo en Nombre del grupo de seguridad (ID). Contendrá el nombre del clúster y tendrá un aspecto similar al siguiente: `docdb-ec2-docdb-2023-12-11-21-33-41:i-0e4bb09985d2bbc4c (sg-0238e0b0bf0f73877)`.

8. En la sección Autenticación, ingrese un nombre de usuario para el usuario principal y, a continuación, elija Autoadministrado. Ingrese una contraseña y, a continuación, confírmela.

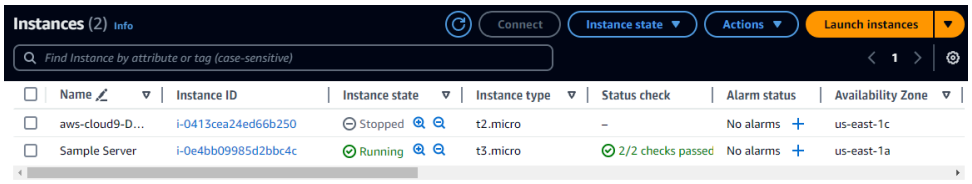
   Si, por el contrario, eligió Administrado en AWS Secrets Manager, consulte Administración de contraseñas con Amazon DocumentDB y AWS Secrets Manager para obtener más información.

9. Elija Create cluster.

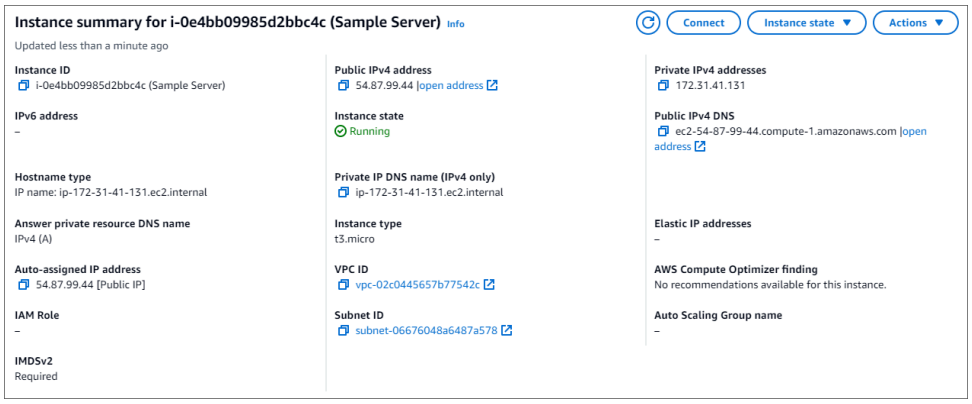
Paso 3: conexión a la instancia de Amazon EC2

Para instalar el intérprete de comandos de mongo, primero se debe conectar a la instancia de Amazon EC2. La instalación del intérprete de comandos de mongo permite conectarse a su clúster de Amazon DocumentDB y realizar consultas en él. Realice los siguientes pasos:

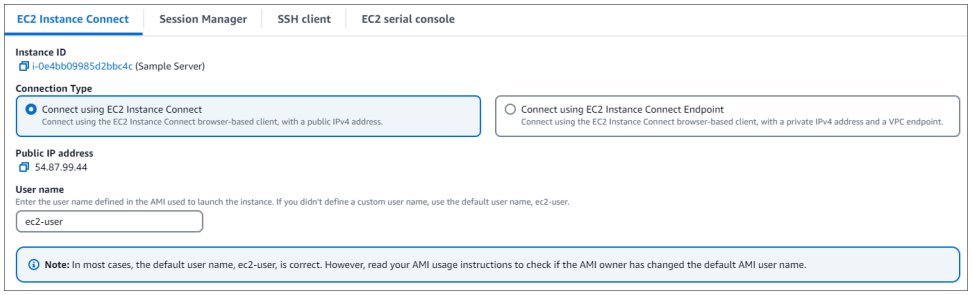
1. En la consola Amazon EC2, navegue hasta sus instancias y compruebe si la instancia que acaba de crear se está ejecutando. Si es así, haga clic en el ID de la instancia para seleccionar la instancia.



2. Elija Conectar.



3. Hay cuatro opciones con pestañas para el método de conexión: Amazon EC2 Instance Connect, Session Manager, cliente SSH o consola serie de EC2. Debe elegir uno y seguir sus instrucciones. Cuando haya terminado, elija Conectar.



> **Note**
>
> Si su dirección IP ha cambiado después de iniciar este tutorial o si va a volver a su entorno más adelante, debe actualizar la regla de entrada del grupo de seguridad `demoEC2` para habilitar el tráfico entrante desde su nueva dirección de API.

Paso 4: instalación del intérprete de comandos de MongoDB

Ahora puede instalar el intérprete de comandos de MongoDB, que es una utilidad de línea de comandos que se utiliza para conectarse al clúster de Amazon DocumentDB y consultarlo. Actualmente hay dos versiones del intérprete de comandos de MongoDB: la versión más reciente, mongosh, y la versión anterior, el intérprete de comandos de mongo.

> **Important**
>
> Existe una limitación conocida en relación con Node.js los controladores anteriores a la versión 6.13.1, que actualmente no son compatibles con la autenticación de identidad de IAM en Amazon DocumentDB. Node.js Los controladores y las herramientas que utilizan el Node.js controlador (por ejemplo, mongosh) deben actualizarse para que utilicen Node.js la versión 6.13.1 o superior del controlador.

Siga las siguientes instrucciones para instalar el intérprete de comandos de MongoDB en su sistema operativo.

On Amazon Linux

Cómo instalar el intérprete de comandos de MongoDB en Amazon Linux

Si no utiliza la autenticación de IAM y desea utilizar el último intérprete de comandos de MongoDB (mongosh) para conectarse a su clúster de Amazon DocumentDB, siga estos pasos:

1. Cree el archivo de repositorio. En la línea de comando de la instancia de EC2 que creó, escriba el siguiente 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. Cuando esté completo, instale mongosh con una de las siguientes opciones de comando en el símbolo del sistema:

   Opción 1: si eligió Amazon Linux 2023 (predeterminado) durante la configuración de Amazon EC2, introduzca este comando:

   ```
   sudo yum install -y mongodb-mongosh-shared-openssl3
   ```

   Opción 2: si eligió Amazon Linux 2 durante la configuración de Amazon EC2, introduzca este comando:

   ```
   sudo yum install -y mongodb-mongosh
   ```

Si utiliza la autenticación de IAM, debe utilizar la versión anterior del intérprete de comandos de MongoDB (5.0) para conectarse al clúster de Amazon DocumentDB. Siga estos pasos:

1. Cree el archivo de repositorio. En la línea de comando de la instancia de EC2 que creó, escriba el siguiente 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. Cuando esté completo, instale el intérprete de comandos de MongoDB 5.0 con la siguiente opción de comando en el símbolo del sistema:

   ```
   sudo yum install -y mongodb-org-shell
   ```

On Ubuntu

Cómo instalar mongosh en Ubuntu

1. Importe la clave pública que utilizará el sistema de administración de paquetes.

```
curl -fsSL https://pgp.mongodb.com/server-5.0.asc | sudo gpg --dearmor -o /usr/
share/keyrings/mongodb-server-5.0.gpg
```

2. Cree el archivo de lista `mongodb-org-5.0.list` para MongoDB mediante el comando correspondiente a su versión de 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. Importe y actualice la base de datos del paquete local utilizando el siguiente comando:

```
sudo apt-get update
```

4. Instale mongosh.

```
sudo apt-get install -y mongodb-mongosh
```

Para obtener información sobre cómo instalar las versiones anteriores de MongoDB en un sistema Ubuntu, consulte Install MongoDB Community Edition on Ubuntu.

On other operating systems

Para instalar el intérprete de comandos de mongo en otros sistemas operativos, consulte el tema sobre cómo instalar MongoDB Community Edition en la documentación de MongoDB.

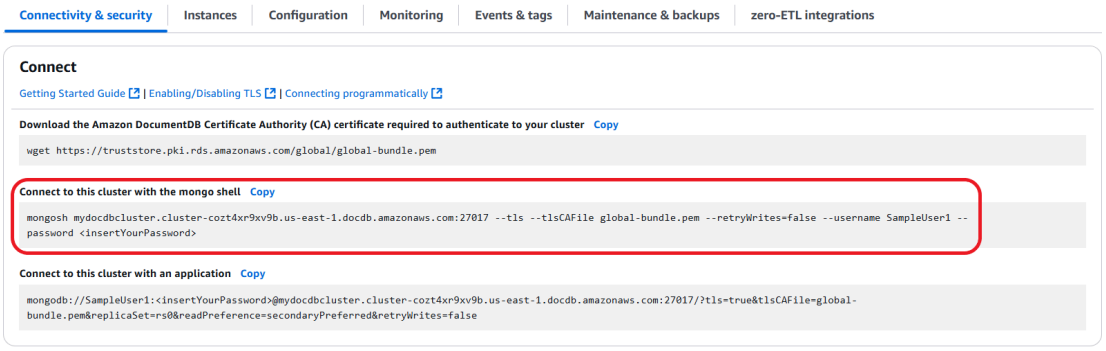
Paso 5: administración de TLS de Amazon DocumentDB

Descargue el certificado CA para Amazon DocumentDB con el siguiente código: `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

> **Note**
>
> La seguridad de la capa de transporte (TLS) está habilitada de forma predeterminada para todos los clústeres nuevos de Amazon DocumentDB. Para obtener más información, consulte Cifrado de datos en tránsito.

Paso 6: conexión al clúster de Amazon DocumentDB

1. En la consola de Amazon DocumentDB, en Clústeres, localice su clúster. Haga clic en Identificador del clúster para elegir el clúster que creó.
2. En la pestaña Conectividad y seguridad, busque Conectar a este clúster con el intérprete de comandos de mongo en el cuadro Conectar:



   Copie la cadena de conexión proporcionada y péguela en el terminal.

   Haga los siguientes cambios en ella:

   a. Asegúrese de que tenga el nombre de usuario correcto en la cadena.
   b. Omita `<insertYourPassword>` para que el intérprete de comandos de mongo le pida la contraseña cuando se conecte.
   c. Opcional: si utiliza la autenticación de IAM o utiliza la versión anterior del intérprete de comandos de MongoDB, modifique la cadena de conexión de la siguiente manera:

      ```
      mongo --ssl --host mydocdbcluster.cluster-cozt4xr9xv9b.us-
      east-1.docdb.amazonaws.com:27017 --sslCAFile global-bundle.pem --
      username SampleUser1 --password
      ```

      Sustituya `mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1` por la misma información de su clúster.
3. Pulse Intro en el terminal. Ahora se le solicitará la contraseña. Introduzca su contraseña.
4. Cuando introduzca la contraseña y pueda ver el mensaje `rs0 [direct: primary] <env-name>>`, significa que se ha conectado correctamente a su clúster de Amazon DocumentDB.

Para resolver los problemas de conexión, consulte. Solución de problemas de Amazon DocumentDB

Paso 7: inserción y consulta de datos

Ahora que está conectado a su clúster, puede realizar algunas consultas para familiarizarse con el uso de una base de datos de documentos.

1. Para insertar un solo documento, escriba lo siguiente:

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

   Obtiene el siguiente resultado:

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

2. Puede leer el documento que escribió con el comando `findOne()` (ya que solo devuelve un documento). La siguiente entrada:

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

   Obtiene el siguiente resultado:

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

3. Para realizar algunas consultas más, plantéese un caso de uso de perfiles de juegos. Primero, inserte algunas entradas en una colección titulada `profiles`. La siguiente entrada:

```
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 }
])
```

   Obtiene el siguiente resultado:

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

4. Utilice el comando `find()` para devolver todos los documentos de la colección de perfiles. La siguiente entrada:

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

   Obtendrá un resultado que coincidirá con los datos que escribió en el paso 3.

5. Utilice una consulta para un único documento mediante un filtro. La siguiente entrada:

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

   Obtiene el siguiente resultado:

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

6. Ahora intentemos buscar un perfil y modificarlo con el comando `findAndModify`. Le daremos al usuario Matt 10 puntos adicionales con el siguiente código:

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

   Se obtiene el siguiente resultado (tenga en cuenta que la puntuación aún no ha aumentado):

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

7. Puede comprobar que su puntuación ha cambiado con la siguiente consulta:

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

   Obtiene el siguiente resultado:

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

Paso 8: exploración

¡Enhorabuena! Ha completado correctamente la Guía de inicio rápido para Amazon DocumentDB.

Pasos siguientes Descubra cómo aprovechar al máximo esta potente base de datos con algunas de sus características más populares:

- Administración de clústeres e instancias de Amazon DocumentDB
- Escalado
- Copia de seguridad y restauración

> **Note**
>
> Para ahorrar costos, puede detener el clúster de Amazon DocumentDB para reducir los costos o eliminar el clúster. De forma predeterminada, tras 30 minutos de inactividad, su AWS Cloud9 entorno detendrá la instancia Amazon EC2 subyacente.

Conexión automática de una instancia de EC2 a una base de datos de Amazon DocumentDB existente

En el siguiente procedimiento se presupone que ya tiene un clúster de Amazon DocumentDB y una instancia de Amazon EC2.

Acceso al clúster de Amazon DocumentDB y configuración de la conexión de Amazon EC2

1. Acceda al clúster de Amazon DocumentDB.

   a. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb

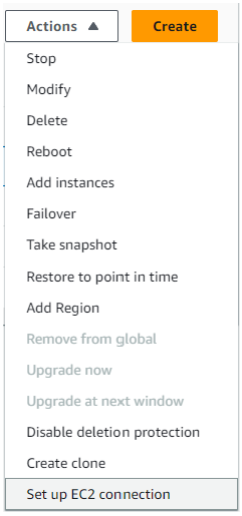
   b. En el panel de navegación, seleccione Clusters (Clústeres).

      > **Tip**
      >
      > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

   c. Para especificar el clúster que desea, seleccione el botón situado a la izquierda del nombre del clúster.

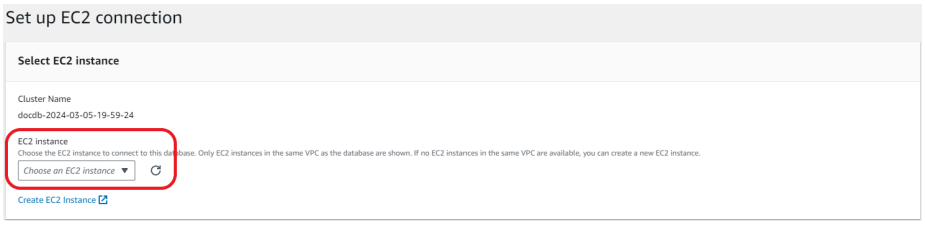
2. Configure la conexión de Amazon EC2.

a. Elija Acciones y, a continuación, elija Configurar conexión de EC2.



Aparece el cuadro de diálogo Configurar conexión de EC2.

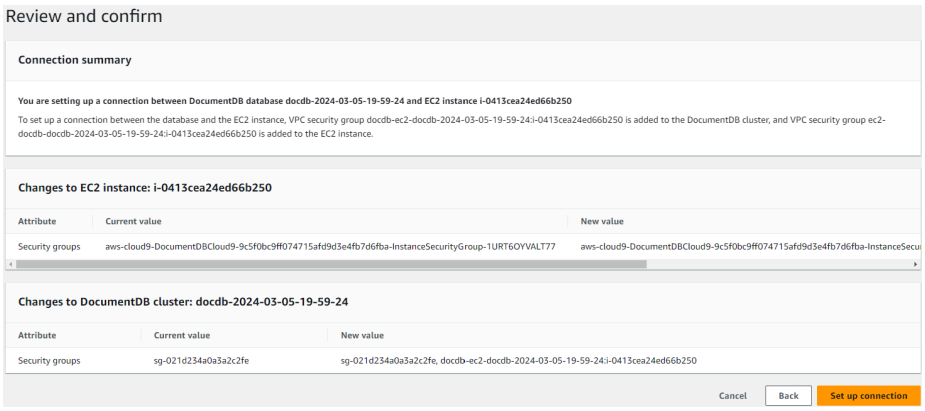
b. En el campo Instancia de EC2, elija la instancia de EC2 que desea conectar al clúster.



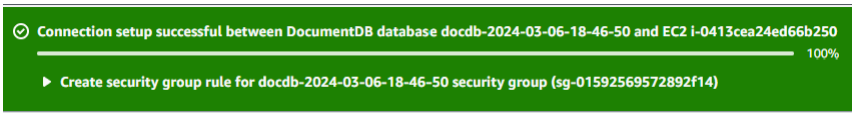
c. Elija Continuar.

Aparece el cuadro de diálogo Revisar y confirmar.

d. Asegúrese de que los cambios sean correctos. A continuación, elija Configurar la conexión.



Si la operación se lleva a cabo correctamente, aparecerá la siguiente verificación:



Descripción general de la conectividad automática con una instancia de EC2

Cuando se configura una conexión entre una instancia de EC2 y una base de datos de Amazon DocumentDB, Amazon DocumentDB configura automáticamente el grupo de seguridad de la VPC para la instancia de EC2 y la base de datos de Amazon DocumentDB.

Estos son los requisitos para conectar una instancia de EC2 a una base de datos de Amazon DocumentDB:

- La instancia de EC2 debe existir en la misma VPC que la base de datos de Amazon DocumentDB.

  Si no existen instancias de EC2 en la misma VPC, la consola proporciona un enlace para crear una.
- El usuario que establece la conectividad debe tener permisos para realizar las siguientes operaciones de Amazon EC2:
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeSecurityGroups`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

Si la instancia de la base de datos y la instancia de EC2 se encuentran en diferentes zonas de disponibilidad, su cuenta podría incurrir en costos cruzados de la zona de disponibilidad.

Cuando se establece una conexión con una instancia de EC2, Amazon DocumentDB lleva a cabo una acción basada en la configuración actual de los grupos de seguridad asociados a la base de datos de Amazon DocumentDB y la instancia de EC2, como se describe en la siguiente tabla:

Configuración actual del grupo de seguridad de Amazon DocumentDB	Configuración del grupo de seguridad de EC2 actual	Acción de Amazon DocumentDB
Hay uno o más grupos de seguridad asociados a la base de datos de Amazon DocumentDB con un nombre que coincide con el patrón `DocumentDB-ec2-n` . No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad tiene solo una regla de entrada con el grupo de seguridad de VPC de la instancia de EC2 como origen.	Hay uno o más grupos de seguridad asociados a la instancia de EC2 con un nombre que coincide con el patrón `DocumentDB-ec2-n` (donde n es un número). No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad solo tiene una regla de salida con el grupo de seguridad de VPC de la base de datos de Amazon DocumentDB como origen.	Amazon DocumentDB no lleva a cabo ninguna acción. Ya se configuró automátic amente una conexión entre la instancia de EC2 y la base de datos de Amazon DocumentDB. Como ya existe una conexión entre la instancia de EC2 y la base de datos de Amazon DocumentD B, los grupos de seguridad no se modifican.
Se aplica alguna de las siguientes condiciones: • No hay ningún grupo de seguridad asociado a la base de datos de Amazon DocumentDB con un nombre que coincide con el patrón `DocumentDB-ec2-n`. • Hay uno o más grupos de seguridad asociados a Amazon DocumentDB con un nombre que coincide con el patrón `DocumentD B-ec2-n` . Sin embargo, Amazon DocumentDB no puede usar ninguno de	Se aplica alguna de las siguientes condiciones: • No hay ningún grupo de seguridad asociado a la instancia de EC2 con un nombre que coincida con el patrón `ec2-Docum entDB-n` . • Hay uno o más grupos de seguridad asociados a la instancia de EC2 con un nombre que coincide con el patrón `ec2-Docum entDB-n` . Sin embargo, Amazon DocumentDB no puede usar ninguno de estos grupos de seguridad	Acción de Amazon DocumentDB: creación de nuevos grupos de seguridad

Configuración actual del grupo de seguridad de Amazon DocumentDB	Configuración del grupo de seguridad de EC2 actual	Acción de Amazon DocumentDB
estos grupos de seguridad para la conexión con la instancia de EC2. Amazon DocumentDB no puede usar un grupo de seguridad que no tenga una regla de entrada con el grupo de seguridad de la VPC de la instancia de EC2 como origen. Amazon DocumentDB tampoco puede usar un grupo de seguridad que se haya modificado. Los ejemplos de modificaciones incluyen agregar una regla o cambiar el puerto de una regla existente.	para la conexión con la base de datos de Amazon DocumentDB. Amazon DocumentDB no puede usar un grupo de seguridad que no tenga una regla de salida con el grupo de seguridad de la VPC de la base de datos de Amazon DocumentDB como origen. Amazon DocumentDB tampoco puede usar un grupo de seguridad que se haya modificado.	

Configuración actual del grupo de seguridad de Amazon DocumentDB	Configuración del grupo de seguridad de EC2 actual	Acción de Amazon DocumentDB
Hay uno o más grupos de seguridad asociados a la base de datos de Amazon DocumentDB con un nombre que coincide con el patrón `DocumentDB-ec2-n` . No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad tiene solo una regla de entrada con el grupo de seguridad de VPC de la instancia de EC2 como origen.	Hay uno o más grupos de seguridad asociados a la instancia de EC2 con un nombre que coincide con el patrón `ec2-DocumentDB-n` . Sin embargo, Amazon DocumentDB no puede usar ninguno de estos grupos de seguridad para la conexión con la base de datos de Amazon DocumentDB. Amazon DocumentDB no puede usar un grupo de seguridad que no tenga una regla de salida con el grupo de seguridad de la VPC de la base de datos de Amazon DocumentDB como origen. Amazon DocumentDB tampoco puede usar un grupo de seguridad que se haya modificado.	Acción de Amazon DocumentDB: creación de nuevos grupos de seguridad

Configuración actual del grupo de seguridad de Amazon DocumentDB	Configuración del grupo de seguridad de EC2 actual	Acción de Amazon DocumentDB
Hay uno o más grupos de seguridad asociados a la base de datos de Amazon DocumentDB con un nombre que coincide con el patrón `DocumentDB-ec2-n` . No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad tiene solo una regla de entrada con el grupo de seguridad de VPC de la instancia de EC2 como origen.	Existe un grupo de seguridad de EC2 válido para la conexión, pero no está asociado a la instancia de EC2. Este grupo de seguridad tiene un nombre que coincide con el patrón `DocumentDB-ec2-n`. No se ha modificado. Solo tiene una regla de salida con el grupo de seguridad de VPC de la base de datos de Amazon DocumentDB como origen.	Acción de Amazon DocumentDB: asociación de un grupo de seguridad de EC2

Configuración actual del grupo de seguridad de Amazon DocumentDB	Configuración del grupo de seguridad de EC2 actual	Acción de Amazon DocumentDB
Se aplica alguna de las siguientes condiciones: • No hay ningún grupo de seguridad asociado a la base de datos de Amazon DocumentDB con un nombre que coincide con el patrón `DocumentDB-ec2-n`. • Hay uno o más grupos de seguridad asociados a la base de datos de Amazon DocumentDB con un nombre que coincide con el patrón `DocumentDB-ec2-n` . Sin embargo, Amazon DocumentDB no puede usar ninguno de estos grupos de seguridad para la conexión con la instancia de EC2. Amazon DocumentDB no puede usar un grupo de seguridad que no tenga una regla de entrada con el grupo de seguridad de la VPC de la instancia de EC2 como origen. Amazon DocumentDB tampoco puede usar un grupo de seguridad modificado.	Hay uno o más grupos de seguridad asociados a la instancia de EC2 con un nombre que coincide con el patrón `DocumentDB-ec2-n`. No se ha modificad o ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad solo tiene una regla de salida con el grupo de seguridad de VPC de la base de datos de Amazon DocumentDB como origen.	Acción de Amazon DocumentDB: creación de nuevos grupos de seguridad

Acción de Amazon DocumentDB: creación de nuevos grupos de seguridad

Amazon DocumentDB lleva a cabo las siguientes acciones:

- Crea un nuevo grupo de seguridad que coincide con el patrón `DocumentDB-ec2-n`. Este grupo de seguridad tiene una regla de entrada con el grupo de seguridad de VPC de la instancia de EC2 como origen. Este grupo de seguridad está asociado a la base de datos de Amazon DocumentDB y permite que la instancia de EC2 acceda a la base de datos de Amazon DocumentDB.
- Crea un nuevo grupo de seguridad que coincide con el patrón `ec2-DocumentDB-n`. Este grupo de seguridad tiene una regla de salida con el grupo de seguridad de VPC de la base de datos de Amazon DocumentDB como origen. Este grupo de seguridad está asociado a la instancia de EC2 y permite que la instancia de EC2 envíe tráfico a la base de datos de Amazon DocumentDB.

Acción de Amazon DocumentDB: asociación de un grupo de seguridad de EC2

Amazon DocumentDB asocia el grupo de seguridad de EC2 válido y existente con la instancia de EC2. Este grupo de seguridad permite que la instancia de EC2 envíe tráfico a la base de datos de Amazon DocumentDB.

Visualización de los recursos de computación conectados

Puede utilizarla Consola de administración de AWS para ver los recursos informáticos que están conectados a una base de datos de Amazon DocumentDB. Los recursos que se muestran incluyen conexiones de recursos informáticos que se configuraron automáticamente. Puede definir la conectividad con los recursos informáticos de manera automática de las siguientes maneras:

- Puede seleccionar el recurso informático al crear la base de datos. Para obtener más información, consulte Creación de un clúster de Amazon DocumentDB Creación de un Multi-AZ clúster de base de datos.
- Puede configurar la conectividad entre una base de datos existente y un recurso informático. Para obtener más información, consulte Conexión automática de Amazon EC2.

Los recursos informáticos de la lista no incluyen los que se conectaron a la base de datos manualmente. Por ejemplo, puede permitir que un recurso informático acceda a una base de datos manualmente añadiendo una regla al grupo de seguridad de la VPC asociado a la base de datos.

Para que un recurso informático coincida, se deben cumplir las siguientes condiciones:

- El nombre del grupo de seguridad asociado al recurso de computación coincide con el patrón `ec2-DocumentDB-n` (donde n es un número).
- El grupo de seguridad asociado al recurso de computación tiene una regla de salida con el rango de puertos establecido en el puerto utilizado por la base de datos de Amazon DocumentDB.
- El grupo de seguridad asociado al recurso de computación tiene una regla de salida en la que el origen está establecido en un grupo de seguridad asociado a la base de datos de Amazon DocumentDB.
- El nombre del grupo de seguridad asociado a la base de datos de Amazon DocumentDB coincide con el patrón `DocumentDB-ec2-n` (donde n es un número).
- El grupo de seguridad asociado a la base de datos de Amazon DocumentDB tiene una regla de entrada con el rango de puertos establecido en el puerto utilizado por la base de datos de Amazon DocumentDB.
- El grupo de seguridad asociado a la base de datos de Amazon DocumentDB tiene una regla de entrada con el origen establecido en un grupo de seguridad asociado al recurso de computación.

Visualización de los recursos de computación que están conectados a una base de datos de Amazon DocumentDB

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, elija Bases de datos y, luego, el nombre de la instancia de base de datos de Amazon DocumentDB.
3. En la pestaña Conectividad y seguridad, consulte los recursos de computación en la sección Recursos de computación conectados.

Conexión manual de Amazon EC2

Temas

- Paso 1: creación de una instancia de Amazon EC2
- Paso 2: crear un grupo de seguridad
- Paso 3: crear un clúster de Amazon DocumentDB
- Paso 4: conectarse a su instancia de Amazon EC2
- Paso 5: instalación del intérprete de comandos de MongoDB
- Paso 6: administrar TLS de Amazon DocumentDB

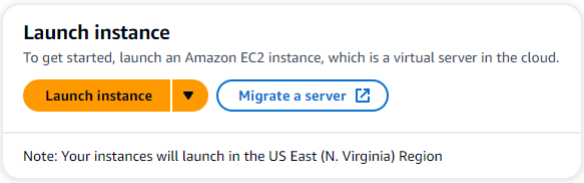
- Paso 7: conectarse a su clúster de Amazon DocumentDB
- Paso 8: insertar y consultar datos
- Paso 9: explorar

En los siguientes pasos se presupone que ha completado los pasos que se detallan en el tema Requisitos previos.

Paso 1: creación de una instancia de Amazon EC2

En este paso, creará una instancia de Amazon EC2 en la misma región y Amazon VPC que utilizará más adelante para aprovisionar el clúster de Amazon DocumentDB.

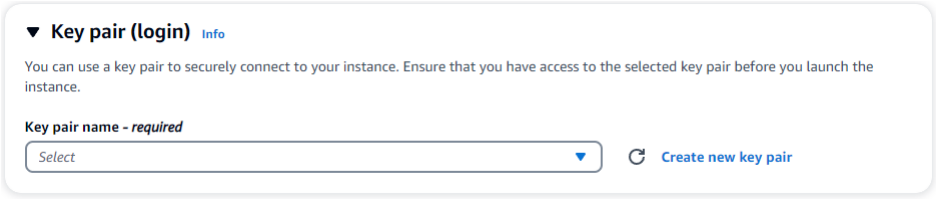
1. En el panel de la consola de Amazon EC2, seleccione Lanzar instancia.



2. Ingrese un nombre o identificador en el campo Nombre ubicado en la sección Nombres y etiquetas.
3. En la lista desplegable Imagen de máquina de Amazon (AMI), busque la AMI de Amazon Linux 2 y selecciónela.



4. Busque y seleccione t3.micro en la lista desplegable de Tipo de instancia.
5. En la sección Par de claves (inicio de sesión), ingrese el identificador de un par de claves existente o elija Crear un par de claves nuevo.



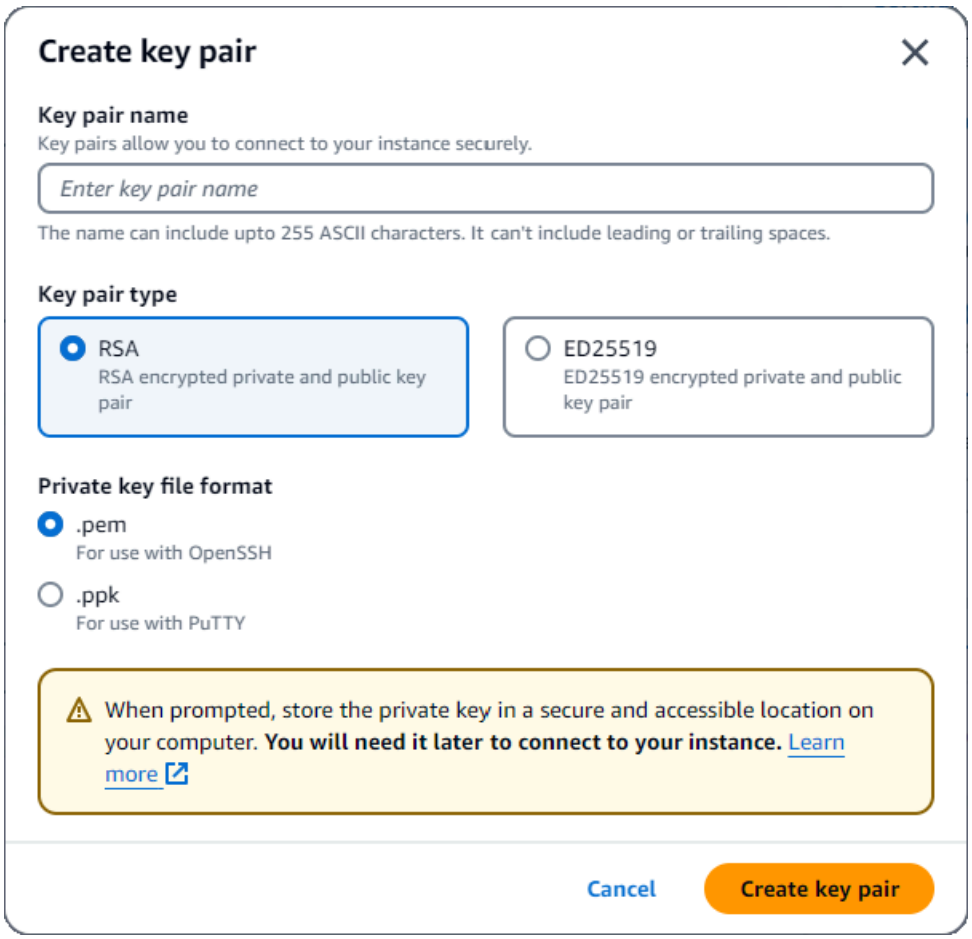
También debe proporcionar un par de claves de Amazon EC2.

- Si tiene un par de claves de Amazon EC2:
  1. Seleccione un par de claves, elija su par de claves de la lista.

2. Debe tener el archivo de clave privada (archivo .pem o .ppk) a su disposición para poder iniciar sesión en la instancia de Amazon EC2.

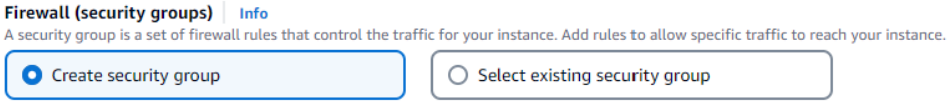
- Si no tiene un par de claves de Amazon EC2:

1. Seleccione Crear un par de claves nuevo y aparecerá el cuadro de diálogo Crear par de claves.
2. Escriba un nombre en el campo Nombre del par de claves.
3. Seleccione el Tipo de par de claves y el Formato de archivo de clave privada.
4. Elija Crear par de claves.



> **Note**
>
> Por motivos de seguridad, le recomendamos encarecidamente que utilice un par de claves para la conectividad a Internet y SSH con la instancia de EC2.

6. En la sección Configuración de red, en Firewall (grupos de seguridad), elija Crear grupo de seguridad o Seleccionar grupo de seguridad existente.



   Si opta por seleccionar un grupo de seguridad existente, seleccione uno en la lista desplegable de Grupos de seguridad comunes.

   Si ha optado por crear un grupo de seguridad nuevo, haga lo siguiente:

   a. Compruebe todas las reglas de tráfico que se aplican a la conectividad de EC2.

   b. En el campo IP, elija Mi IP o seleccione Personalizado para elegir entre una lista de bloques de CIDR, listas de prefijos o grupos de seguridad. No recomendamos Cualquier lugar como opción, a menos que la instancia de EC2 se encuentre en una red aislada, ya que permite el acceso a la instancia de EC2 desde cualquier dirección IP.

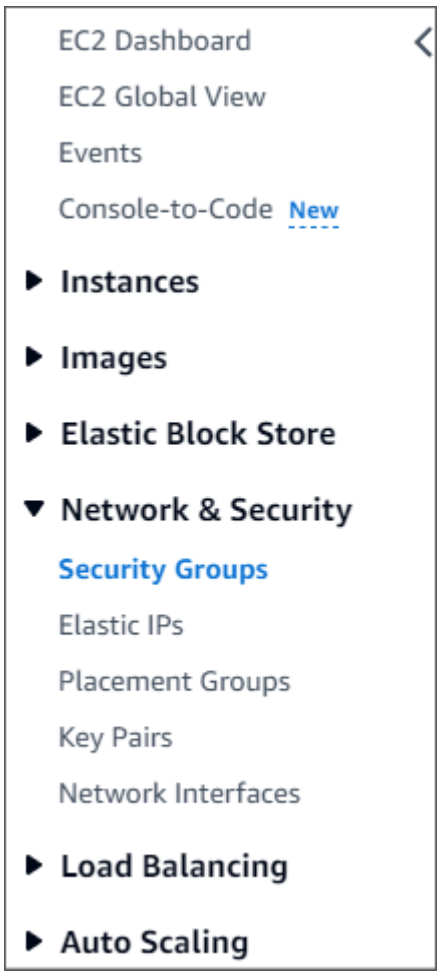


7. En la sección Resumen, revise la configuración de EC2 y, si es correcta, elija Lanzar instancia.

Paso 2: crear un grupo de seguridad

Ahora creará un nuevo grupo de seguridad en su Amazon VPC predeterminada. El grupo de seguridad `demoDocDB` le permite conectarse a su clúster de Amazon DocumentDB en el puerto 27017 (el puerto predeterminado de Amazon DocumentDB) desde su instancia de Amazon EC2.

1. En la Consola de administración de Amazon EC2, en Red y seguridad, elija Grupos de seguridad.



2. Elija Creación de grupo de seguridad.



3. En la sección Detalles básicos:

   a. En Nombre del grupo de seguridad, introduzca `demoDocDB`.

   b. En Descripción, escriba una descripción.

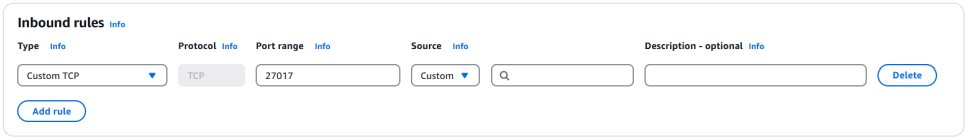
   c. En VPC, acepte el uso de la VPC predeterminada.

4. En la sección Inbound rules (Reglas de entrada), elija Add rule (agregar regla).

   a. En Tipo, elija Regla TCP personalizada (predeterminada).

   b. En Rango de puertos, introduzca `27017`.

   c. En Source (Origen), seleccione Custom (Personalizado). En el campo contiguo, busque el grupo de seguridad que acaba de crear en el paso 1. Es posible que tenga que actualizar el navegador para que la consola Amazon EC2 complete automáticamente el nombre de la fuente .



5. Acepte todos los demás valores predeterminados y elija Crear grupo de seguridad.



Paso 3: crear un clúster de Amazon DocumentDB

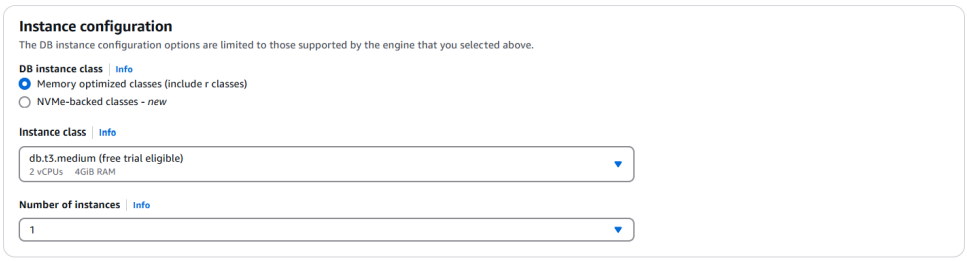
Mientras se aprovisiona la instancia de Amazon EC2, creará su clúster de Amazon DocumentDB.

1. Navegue hasta la consola de Amazon DocumentDB y elija Clústeres en el panel de navegación.
2. Seleccione Crear.
3. Deje la configuración de Tipo de clúster como predeterminada en Clúster basado en instancias.
4. En Configuración del clúster, ingrese un nombre único en Identificador de clúster. Tenga en cuenta que la consola cambiará todos los nombres de los clústeres a minúsculas independientemente de cómo se ingresen.

   Deje Versión del motor con su valor predeterminado de 5.0.0.
5. En Configuración del almacenamiento en clúster, deje la configuración predeterminada de Amazon DocumentDB estándar.
6. En Configuración de instancias:

   - En Clase de instancia de base de datos, elija Clases optimizadas para memoria (incluidas las clases r), que es la opción predeterminada.

     La otra opción de instancia son NVMe-backed las clases. Para obtener más información, consulte NVMe-backed instancias.
   - En Clase de instancia, elija el tipo de instancia que se adapte a sus necesidades. Para obtener una explicación más detallada de las clases de instancia, consulte Especificaciones de clases de instancias.
   - En Número de instancias, elija el número que mejor refleje sus necesidades. Recuerde que cuanto menor sea el número, menor será el costo y menor será el read/write volumen que puede administrar el clúster.



7. En Conectividad, deje la configuración predeterminada de No conectarse a un recurso de computación de EC2.

> **Note**
>
> La conexión a un recurso de computación de EC2 crea automáticamente grupos de seguridad para la conexión al clúster. Como en el paso anterior creó estos grupos de seguridad manualmente, debe seleccionar No conectarse a un recurso de computación de EC2 para no crear un segundo conjunto de grupos de seguridad.

8. En la sección Autenticación, ingrese un nombre de usuario para el usuario principal y, a continuación, elija Autoadministrado. Ingrese una contraseña y, a continuación, confírmela.

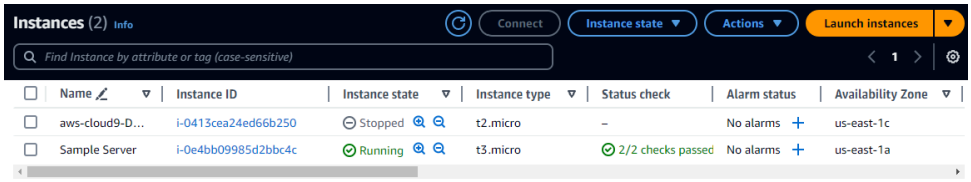
   Si, por el contrario, eligió Administrado en AWS Secrets Manager, consulte Administración de contraseñas con Amazon DocumentDB y AWS Secrets Manager para obtener más información.

9. Elija Create cluster.

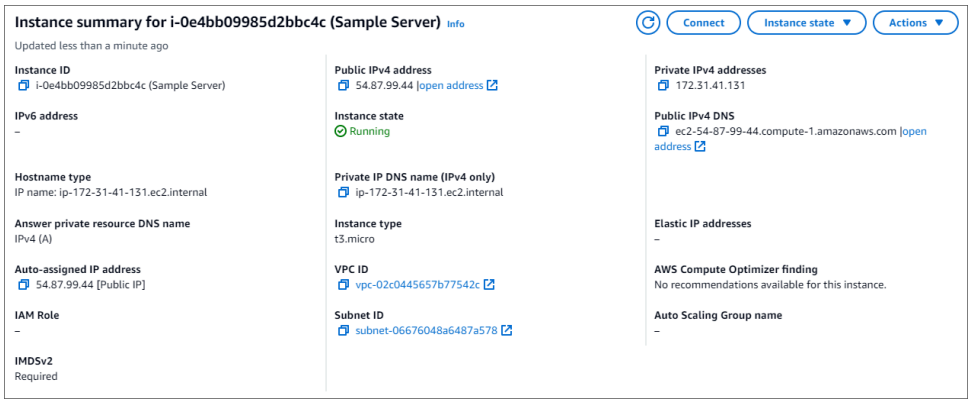
Paso 4: conectarse a su instancia de Amazon EC2

La conexión de la instancia de Amazon EC2 le permitirá instalar el intérprete de comandos de MongoDB. La instalación del intérprete de comandos de mongo permite conectarse a su clúster de Amazon DocumentDB y realizar consultas en él. Realice los siguientes pasos:

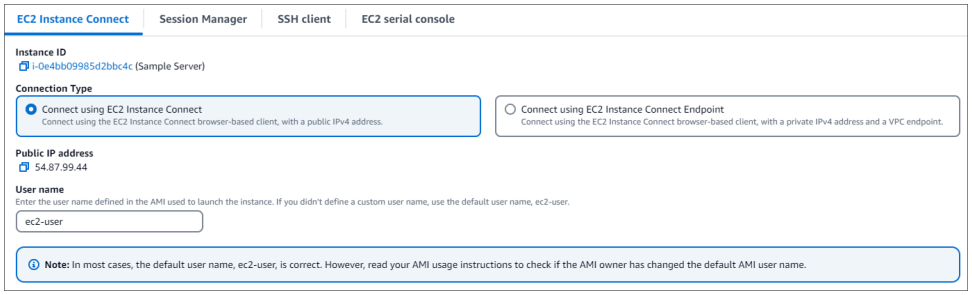
1. En la consola Amazon EC2, navegue hasta sus instancias y compruebe si la instancia que acaba de crear se está ejecutando. Si es así, haga clic en el ID de la instancia para seleccionar la instancia.



2. Elija Conectar.



3. Hay cuatro opciones con pestañas para el método de conexión: Amazon EC2 Instance Connect, Session Manager, cliente SSH o consola serie de EC2. Debe elegir uno y seguir sus instrucciones. Cuando haya terminado, elija Conectar.



> **Note**
>
> Si su dirección IP ha cambiado después de iniciar este tutorial o si va a volver a su entorno más adelante, debe actualizar la regla de entrada del grupo de seguridad `demoEC2` para habilitar el tráfico entrante desde su nueva dirección de API.

Paso 5: instalación del intérprete de comandos de MongoDB

Ahora puede instalar el intérprete de comandos de MongoDB, que es una utilidad de línea de comandos que se utiliza para conectarse al clúster de Amazon DocumentDB y consultarlo. Actualmente hay dos versiones del intérprete de comandos de MongoDB: la versión más reciente, mongosh, y la versión anterior, el intérprete de comandos de mongo.

> **Important**
>
> Existe una limitación conocida en relación con Node.js los controladores anteriores a la versión 6.13.1, que actualmente no son compatibles con la autenticación de identidad de IAM en Amazon DocumentDB. Node.js Los controladores y las herramientas que utilizan el Node.js controlador (por ejemplo, mongosh) deben actualizarse para que utilicen Node.js la versión 6.13.1 o superior del controlador.

Siga las siguientes instrucciones para instalar el intérprete de comandos de MongoDB en su sistema operativo.

On Amazon Linux

Cómo instalar el intérprete de comandos de MongoDB en Amazon Linux

Si no utiliza IAM y desea utilizar el último intérprete de comandos de MongoDB (mongosh) para conectarse a su clúster de Amazon DocumentDB, siga estos pasos:

1. Cree el archivo de repositorio. En la línea de comando de la instancia de EC2 que creó, escriba el siguiente 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. Cuando esté completo, instale mongosh con una de las siguientes opciones de comando en el símbolo del sistema:

   Opción 1: si eligió Amazon Linux 2023 (predeterminado) durante la configuración de Amazon EC2, introduzca este comando:

```
sudo yum install -y mongodb-mongosh-shared-openssl3
```

   Opción 2: si eligió Amazon Linux 2 durante la configuración de Amazon EC2, introduzca este comando:

```
sudo yum install -y mongodb-mongosh
```

Si utiliza IAM, debe utilizar la versión anterior del intérprete de comandos de MongoDB (5.0) para conectarse al clúster de Amazon DocumentDB. Siga estos pasos:

1. Cree el archivo de repositorio. En la línea de comando de la instancia de EC2 que creó, escriba el siguiente 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. Cuando esté completo, instale el intérprete de comandos de MongoDB 5.0 con la siguiente opción de comando en el símbolo del sistema:

```
sudo yum install -y mongodb-org-shell
```

On Ubuntu

Cómo instalar mongosh en Ubuntu

1. Importe la clave pública que utilizará el sistema de administración de paquetes.

```
curl -fsSL https://pgp.mongodb.com/server-5.0.asc | sudo gpg --dearmor -o /usr/
share/keyrings/mongodb-server-5.0.gpg
```

2. Cree el archivo de lista `mongodb-org-5.0.list` para MongoDB mediante el comando correspondiente a su versión de 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. Importe y actualice la base de datos del paquete local utilizando el siguiente comando:

```
sudo apt-get update
```

4. Instale mongosh.

```
sudo apt-get install -y mongodb-mongosh
```

Para obtener información sobre cómo instalar las versiones anteriores de MongoDB en un sistema Ubuntu, consulte Install MongoDB Community Edition on Ubuntu.

On other operating systems

Para instalar el intérprete de comandos de mongo en otros sistemas operativos, consulte el tema sobre cómo instalar MongoDB Community Edition en la documentación de MongoDB.

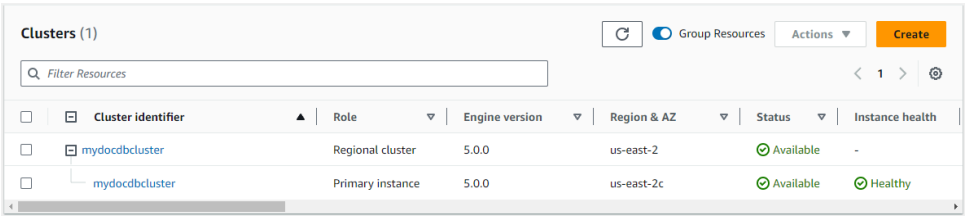
Paso 6: administrar TLS de Amazon DocumentDB

Descargue el certificado CA para Amazon DocumentDB con el siguiente código: `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

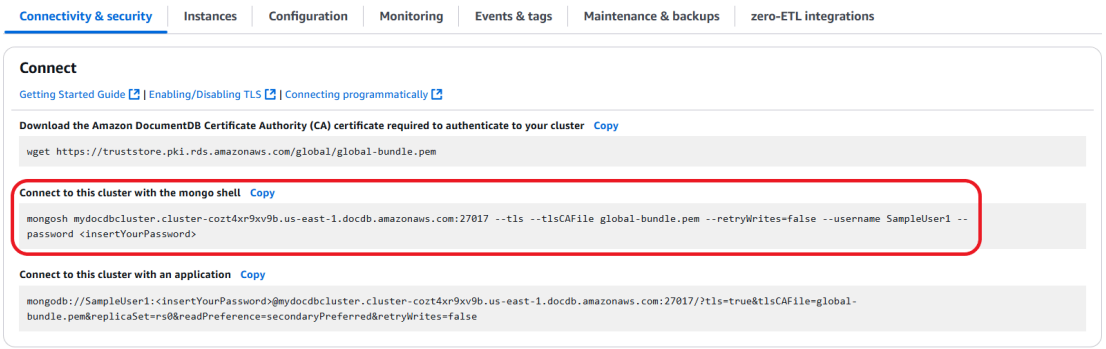
> **Note**
>
> La seguridad de la capa de transporte (TLS) está habilitada de forma predeterminada para todos los clústeres nuevos de Amazon DocumentDB. Para obtener más información, consulte Cifrado de datos en tránsito.

Paso 7: conectarse a su clúster de Amazon DocumentDB

1. En la consola de Amazon DocumentDB, en Clústeres, localice su clúster. Haga clic en Identificador del clúster para elegir el clúster que creó.



2. En la pestaña Conectividad y seguridad, busque Conectar a este clúster con el intérprete de comandos de mongo en el cuadro Conectar:



Copie la cadena de conexión proporcionada y péguela en el terminal.

Haga los siguientes cambios en ella:

a. Asegúrese de que tenga el nombre de usuario correcto en la cadena.

b. Omita `<insertYourPassword>` para que el intérprete de comandos de mongo le pida la contraseña cuando se conecte.

c. Opcional: si utiliza la autenticación de IAM o utiliza la versión anterior del intérprete de comandos de MongoDB, modifique la cadena de conexión de la siguiente manera:

```
mongo --ssl --host mydocdbcluster.cluster-cozt4xr9xv9b.us-
east-1.docdb.amazonaws.com:27017 --sslCAFile global-bundle.pem --
username SampleUser1 --password
```

Sustituya `mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1` por la misma información de su clúster.

3. Pulse Intro en el terminal. Ahora se le solicitará la contraseña. Introduzca su contraseña.

4. Cuando introduzca la contraseña y pueda ver el mensaje `rs0 [direct: primary] <env-name>>`, significa que se ha conectado correctamente a su clúster de Amazon DocumentDB.

Para resolver los problemas de conexión, consulte. Solución de problemas de Amazon DocumentDB

Paso 8: insertar y consultar datos

Ahora que está conectado a su clúster, puede realizar algunas consultas para familiarizarse con el uso de una base de datos de documentos.

1. Para insertar un solo documento, escriba lo siguiente:

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

Obtiene el siguiente resultado:

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

2. Puede leer el documento que escribió con el comando `findOne()` (ya que solo devuelve un documento). La siguiente entrada:

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

Obtiene el siguiente resultado:

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

3. Para realizar algunas consultas más, plantéese un caso de uso de perfiles de juegos. Primero, inserte algunas entradas en una colección titulada `profiles`. La siguiente entrada:

```
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 }
])
```

Obtiene el siguiente resultado:

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

4. Utilice el comando `find()` para devolver todos los documentos de la colección de perfiles. La siguiente entrada:

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

Obtendrá un resultado que coincidirá con los datos que escribió en el paso 3.

5. Utilice una consulta para un único documento mediante un filtro. La siguiente entrada:

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

Obtiene el siguiente resultado:

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

6. Ahora intentemos buscar un perfil y modificarlo con el comando `findAndModify`. Le daremos al usuario Matt 10 puntos adicionales con el siguiente código:

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

Se obtiene el siguiente resultado (tenga en cuenta que la puntuación aún no ha aumentado):

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

7. Puede comprobar que su puntuación ha cambiado con la siguiente consulta:

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

   Obtiene el siguiente resultado:

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

Paso 9: explorar

¡Enhorabuena! Ha completado correctamente la Guía de inicio rápido para Amazon DocumentDB.

Pasos siguientes Descubra cómo aprovechar al máximo esta potente base de datos con algunas de sus características más populares:

- Administración de clústeres e instancias de Amazon DocumentDB
- Escalado
- Copia de seguridad y restauración

> **Note**
>
> Para ahorrar costos, puede detener el clúster de Amazon DocumentDB para reducir los costos o eliminar el clúster. De forma predeterminada, tras 30 minutos de inactividad, su AWS Cloud9 entorno detendrá la instancia Amazon EC2 subyacente.

Conexión mediante el controlador JDBC de Amazon DocumentDB

El controlador JDBC para Amazon DocumentDB proporciona una SQL-relational interfaz para desarrolladores y permite la conectividad desde herramientas de BI como Tableau y. DbVisualizer

Para obtener información más detallada, consulte la documentación del controlador JDBC de Amazon DocumentDB en. GitHub

Temas

- Introducción
- Conexión a Amazon DocumentDB desde Tableau Desktop
- Conéctese a Amazon DocumentDB desde DbVisualizer
- Generación automática de esquemas JDBC
- Compatibilidad y limitaciones de SQL
- Resolución de problemas

Introducción

Paso 1. Creación de un clúster de Amazon DocumentDB

Si no ha creado un clúster de Amazon DocumentDB, cree uno siguiendo las instrucciones de. Introducción a Amazon DocumentDB

> **Note**
>
> Amazon DocumentDB es un servicio exclusivo de nube privada virtual (VPC). Si se conecta desde una máquina local, fuera de la VPC del clúster, necesitará crear una conexión SSH a una instancia de Amazon EC2. En este caso, lance el clúster siguiendo las instrucciones deConectarse mediante Amazon EC2. Consulte Cómo usar un túnel SSH para conectarse a Amazon DocumentDB para obtener más información sobre los túneles SSH y cuándo podría necesitarlos.

Paso 2. Instalación de JRE o JDK

En función de la aplicación de BI que utilice, es posible que tenga instalada en su ordenador una instalación JRE o JDK de 64 bits, versión 8 o posterior. Puede descargar el Java SE Runtime Environment 8 aquí.

Paso 3. Descarga del controlador JDBC de DocumentDB

Descargue el controlador JDBC de DocumentDB desde aquí. El controlador está empaquetado como un único archivo JAR (por ejemplo, documentdb-jdbc-1.0.0-all.jar).

Paso 4. Cómo usar un túnel SSH para conectarse a Amazon DocumentDB

Los clústeres de Amazon DocumentDB (compatible con MongoDB) se implementan en una instancia de Amazon Virtual Private Cloud (Amazon VPC). Las instancias de Amazon EC2 u otros AWS servicios que estén implementados en la misma Amazon VPC pueden acceder a ellos directamente. Además, las instancias EC2a u otros AWS servicios de diferentes VPC de la misma AWS región o de otras regiones pueden acceder a Amazon DocumentDB mediante la interconexión de VPC.

Puede utilizar la tunelización SSH (también conocida como reenvío de puertos) para acceder a los recursos de Amazon DocumentDB desde fuera de la VPC del clúster. Este será el caso de la mayoría de los usuarios que no ejecuten su aplicación en una máquina virtual de la misma VPC que el clúster de DocumentDB.

Para crear un túnel SSH, necesita una instancia de Amazon EC2 que se ejecute en la misma VPC de Amazon que el clúster de Amazon DocumentDB. Puede usar una instancia EC2 existente en la misma VPC que el clúster o crear una. Si es así, puede configurar un túnel de SSH en el clúster de Amazon DocumentDB `sample-cluster.node.us-east-1.docdb.amazonaws.com` ejecutando el siguiente comando en el equipo local.

```
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
```

La marca -L se utiliza para el reenvío de un puerto local. Este es un requisito previo para conectarse a cualquier herramienta de BI que se ejecute en un cliente externo a su VPC. Una vez realizado el paso anterior, puede pasar a los siguientes con la herramienta de BI que prefiera.

Para obtener más información sobre los túneles SSH, consulte la documentación sobre el uso de un túnel SSH para conectarse a Amazon DocumentDB.

Conexión a Amazon DocumentDB desde Tableau Desktop

Temas

- Adición del controlador JDBC de Amazon DocumentDB
- Conexión a Amazon DocumentDB mediante Tableau - Túnel SSH

Adición del controlador JDBC de Amazon DocumentDB

Para conectar con Amazon DocumentDB desde Tableau Desktop, debe descargar e instalar el controlador JDBC de Amazon DocumentDB y el conector de DocumentDB para Tableau.

1. Descargue el archivo JAR del controlador JDBC de Amazon DocumentDB del repositorio de controladores JDBC de Amazon DocumentDB y cópielo en uno de los siguientes directorios según su sistema operativo:
   - Windows - `C:\Program Files\Tableau\Drivers`
   - MacOS - `~/Library/Tableau/Drivers`
2. Descargue el conector Tableau de DocumentDB (un archivo TACO) del sitio web de Tableau Exchange y cópielo en su directorio My Tableau Repository/Connectors
   - Windows - `C:\Users\[user]\Documents\My Tableau Repository\Connectors`
   - MacOS - `/Users/[user]/Documents/My Tableau Repository/Connectors`

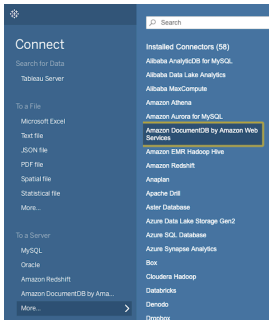
Para obtener información adicional, consulte la documentación de Tableau.

> **Note**
>
> Si utiliza certificados de CA más recientes, asegúrese de actualizar el controlador JDBC a la versión 1.4.5 (disponible en este repositorio). AWS GitHub

Conexión a Amazon DocumentDB mediante Tableau - Túnel SSH

Para conectarse a Tableau desde un equipo cliente fuera de la VPC de su clúster de DocumentDB, debe configurar un túnel SSH antes de seguir los pasos que se indican a continuación:

1. Inicie la aplicación Tableau Desktop.
2. Vaya a Conectar > A un servidor > Más.
3. Elija Amazon DocumentDB de Amazon Web Services en Conectores instalados.

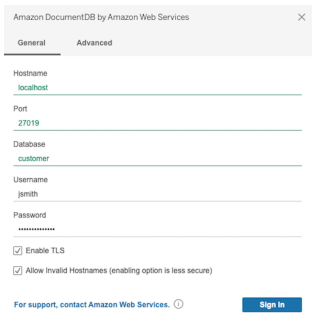


Conexión a Amazon DocumentDB mediante Tableau: túnel SSH externo

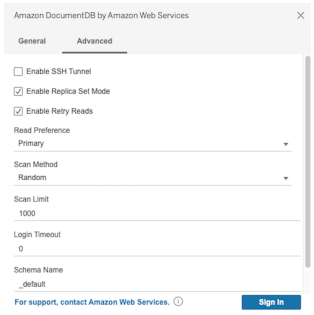
1. Introduzca los parámetros de conexión necesarios: Nombre de host, Puerto, Base de datos, Nombre de usuario y Contraseña. Los parámetros de conexión del siguiente ejemplo son equivalentes a la cadena de conexión JDBC:

   ```
   jdbc:documentdb://localhost:27019/test?
   tls=true&tlsAllowInvalidHostnames=true&scanMethod=random&scanLimit=1000&logir
   ```

   con los parámetros de nombre de usuario y contraseña transferidos por separado en una colección de propiedades. Para obtener más información sobre los parámetros de la cadena de conexión, consulte la documentación de GitHub del controlador JDBC de Amazon DocumentDB.



2. (Opcional) Puede encontrar opciones más avanzadas en la pestaña Avanzado.

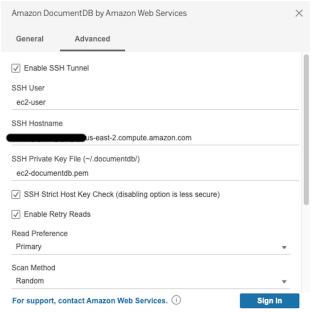


3. Seleccione Iniciar sesión.

Conexión a Amazon DocumentDB mediante Tableau: túnel SSH interno

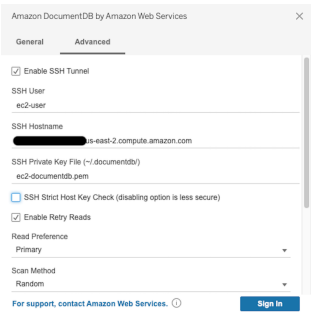
> **Note**
>
> Si prefiere no configurar el túnel SSH mediante un terminal, puede usar la GUI de Tableau para especificar los detalles de su instancia EC2, que el controlador JDBC utilizará de forma inherente para crear un túnel SSH.

1. En la pestaña Avanzado, elija la opción Habilitar el túnel SSH para revisar otras propiedades.



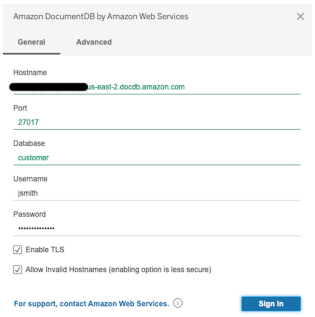
2. Introduzca el Usuario SSH, el Nombre de host SSH y el Archivo de clave privada SSH.
3. (Opcional) Puede deshabilitar la opción de verificación estricta de la clave de host de SSH, que evita la verificación de la clave de host con un archivo de hosts conocido.

> **Note**
>
> Deshabilitar esta opción es menos seguro, ya que puede provocar un ataque de intermediario.



4. Introduzca los parámetros necesarios: Nombre de host, Puerto, Base de datos, Nombre de usuario y Contraseña.

> **Note**
>
> Asegúrese de utilizar el punto de conexión del clúster de DocumentDB y no localhost cuando utilice la opción de túnel SSH interno.



5. Elija Iniciar sesión.

Conéctese a Amazon DocumentDB desde DbVisualizer

Temas

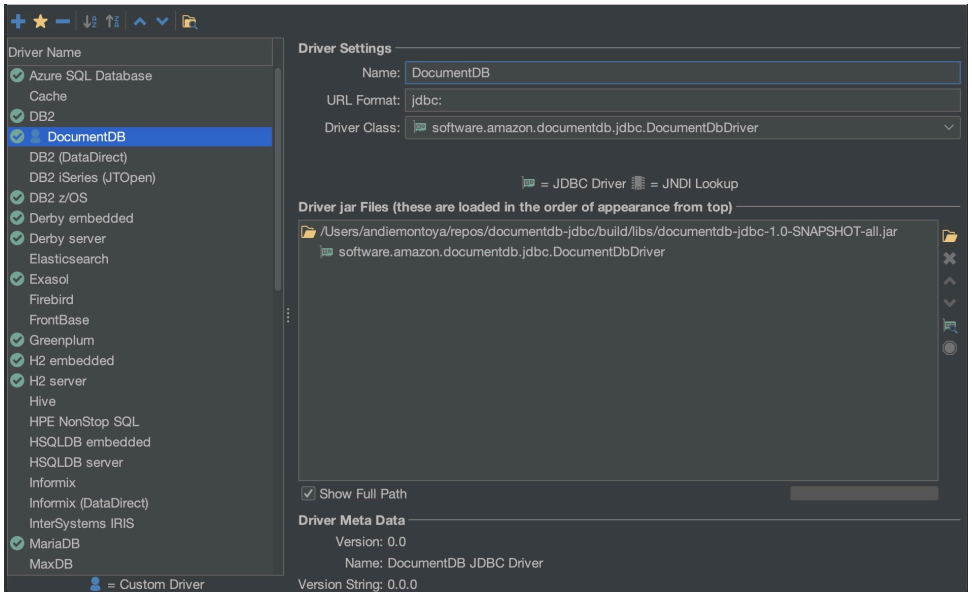
- Adición del controlador JDBC de Amazon DocumentDB
- Conectarse a Amazon DocumentDB mediante DbVisualizer

Adición del controlador JDBC de Amazon DocumentDB

Para conectarse a Amazon DocumentDB desde, primero DbVisualizer debe importar el controlador JDBC de Amazon DocumentDB

1. Inicie la DbVisualizer aplicación y navegue hasta la ruta del menú: Herramientas > Administrador de controladores...
2. Elija + (o en el menú, seleccione Controlador > Crear controlador).
3. Establezca Name (Nombre) en `DocumentDB`.
4. Defina el formato de URL en `jdbc:documentdb://<host>[:port]/<database>[?option=value[&option=value[...]]]`
5. Elija el botón de carpeta y luego seleccione el archivo JAR del controlador JDBC de Amazon DocumentDB y luego pulse el botón Abrir.

6. Compruebe que el campo Clase de controlador esté establecido en `software.amazon.documentdb.jdbc.DocumentDbDriver`. La configuración del Administrador de controladores para DocumentDB debe tener el siguiente aspecto.



7. Cierre el cuadro de diálogo. El controlador JDBC de Amazon DocumentDB estará configurado y listo para usarse.

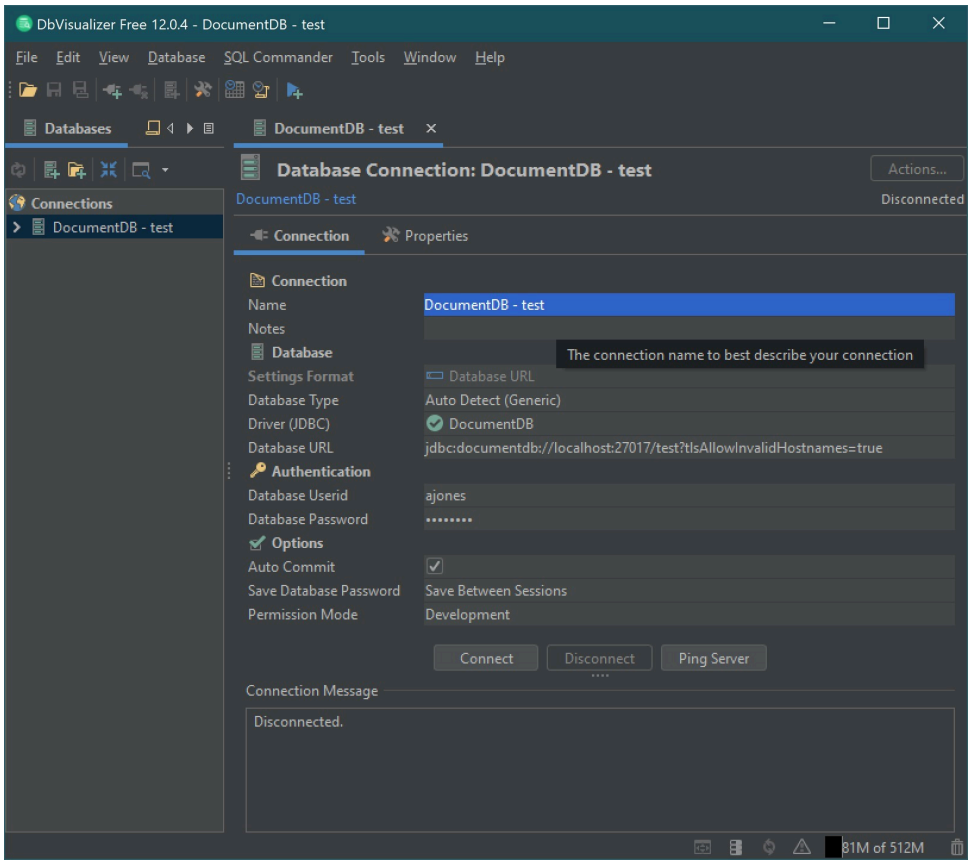
Conectarse a Amazon DocumentDB mediante DbVisualizer

Conéctese a Amazon DocumentDB mediante DbVisualizer

1. Si se conecta desde fuera de la VPC del clúster de Amazon DocumentDB, asegúrese de haber configurado un túnel SSH.
2. Seleccione Base de datos > Crear conexión a base de datos en el menú de nivel superior.
3. En el campo Nombre, ingrese un nombre descriptivo.
4. Establezca Controlador (JDBC) en el controlador de DocumentDB que creó en la sección anterior.
5. Establezca la URL de la base de datos en la cadena de conexión JDBC.

   Por ejemplo: `jdbc:documentdb://localhost:27017/database?tlsAllowInvalidHostnames=true`
6. Establezca el ID de usuario de la base de datos en su ID de usuario de Amazon DocumentDB.
7. Establezca la contraseña de la base de datos como la contraseña correspondiente al ID de usuario.

El cuadro de diálogo de conexión a la base de datos debe tener el siguiente aspecto:



8. Elija Connect (Conectar).

Generación automática de esquemas JDBC

Amazon DocumentDB es una base de datos de documentos y, por lo tanto, no tiene el concepto de tablas y esquema. Sin embargo, las herramientas de inteligencia empresarial, como Tableau, esperan que la base de datos a la que se conecta presente un esquema. En concreto, cuando la conexión del controlador JDBC necesite obtener el esquema de la colección en la base de datos, consultará todas las colecciones de la base de datos. El controlador determinará si ya existe una versión en caché del esquema para esa colección. Si no existe una versión en caché, tomará muestras de la colección de documentos y creará un esquema basado en el siguiente comportamiento.

Temas

- Limitaciones de generación de esquemas
- Opciones del método de análisis

- Tipos de datos de Amazon DocumentDB
- Asignación de campos de documentos escalares
- Manejo de tipos de datos de objetos y matrices

Limitaciones de generación de esquemas

El controlador JDBC de DocumentDB impone un límite de 128 caracteres a la longitud de los identificadores. El generador de esquemas puede truncar la longitud de los identificadores generados (nombres de tablas y nombres de columnas) para garantizar que se ajusten a ese límite.

Opciones del método de análisis

El comportamiento del muestreo se puede modificar mediante la cadena de conexión o las opciones de origen de datos.

- scanMethod=<option>
  - random - (predeterminado): los documentos de muestra se devuelven en orden aleatorio.
  - idForward: los documentos de muestra se devuelven por orden de identificación.
  - idReverse: los documentos de muestra se devuelven por orden inverso de identificación.
  - all (todos): muestra todos los documentos de la colección.
- scanLimit=<n>: el número de documentos que se van a muestrear. El valor debe ser un número entero positivo. El valor predeterminado es 1000. Si ScanMethod está establecido en todos, se omite esta opción.

Tipos de datos de Amazon DocumentDB

El servidor de Amazon DocumentDB admite varios tipos de datos de MongoDB. A continuación se enumeran los tipos de datos compatibles y sus tipos de datos JDBC asociados.

Tipos de datos de MongoDB	Compatible con DocumentDB	Tipo de datos JDBC
Datos Binary	Sí	VARBINARY
Booleano	Sí	BOOLEANO
Double	Sí	DOUBLE

Tipos de datos de MongoDB	Compatible con DocumentDB	Tipo de datos JDBC
Entero de 32 bits	Sí	INTEGER
Entero de 64 bits	Sí	BIGINT
Cadena	Sí	VARCHAR
ObjectId	Sí	VARCHAR
Date	Sí	TIMESTAMP
Nulo	Sí	VARCHAR
Expresión regular	Sí	VARCHAR
Timestamp	Sí	VARCHAR
MinKey	Sí	VARCHAR
MaxKey	Sí	VARCHAR
Objeto	Sí	tabla virtual
Matriz	Sí	tabla virtual
Decimal128	No	DECIMAL
JavaScript	No	VARCHAR
JavaScript (con alcance)	No	VARCHAR
Sin definir	No	VARCHAR
Símbolo	No	VARCHAR
dbPointer (4.0 y versiones posteriores)	No	VARCHAR

Asignación de campos de documentos escalares

Al escanear una muestra de documentos de una colección, el controlador JDBC creará uno o más esquemas para representar las muestras de la colección. En general, un campo escalar del documento se asigna a una columna del esquema de la tabla. Por ejemplo, en una colección denominada team y en un solo documento `{ "_id" : "112233", "name" : "Alastair", "age": 25 }`, esto se asignaría al esquema:

Nombre de la tabla	Nombre de la columna	Tipo de datos	Key
equipo	id de equipo	VARCHAR	PK
equipo	name	VARCHAR	
equipo	edad	INTEGER	

Promoción de conflictos de tipos de datos

Al digitalizar los documentos de muestra, es posible que los tipos de datos de un campo no sean coherentes de un documento a otro. En este caso, el controlador JDBC convertirá el tipo de datos JDBC en un tipo de datos común que se adapte a todos los tipos de datos de los documentos muestreados.

Por ejemplo:

```
{
"_id" : "112233",
"name" : "Alastair", "age" : 25
}

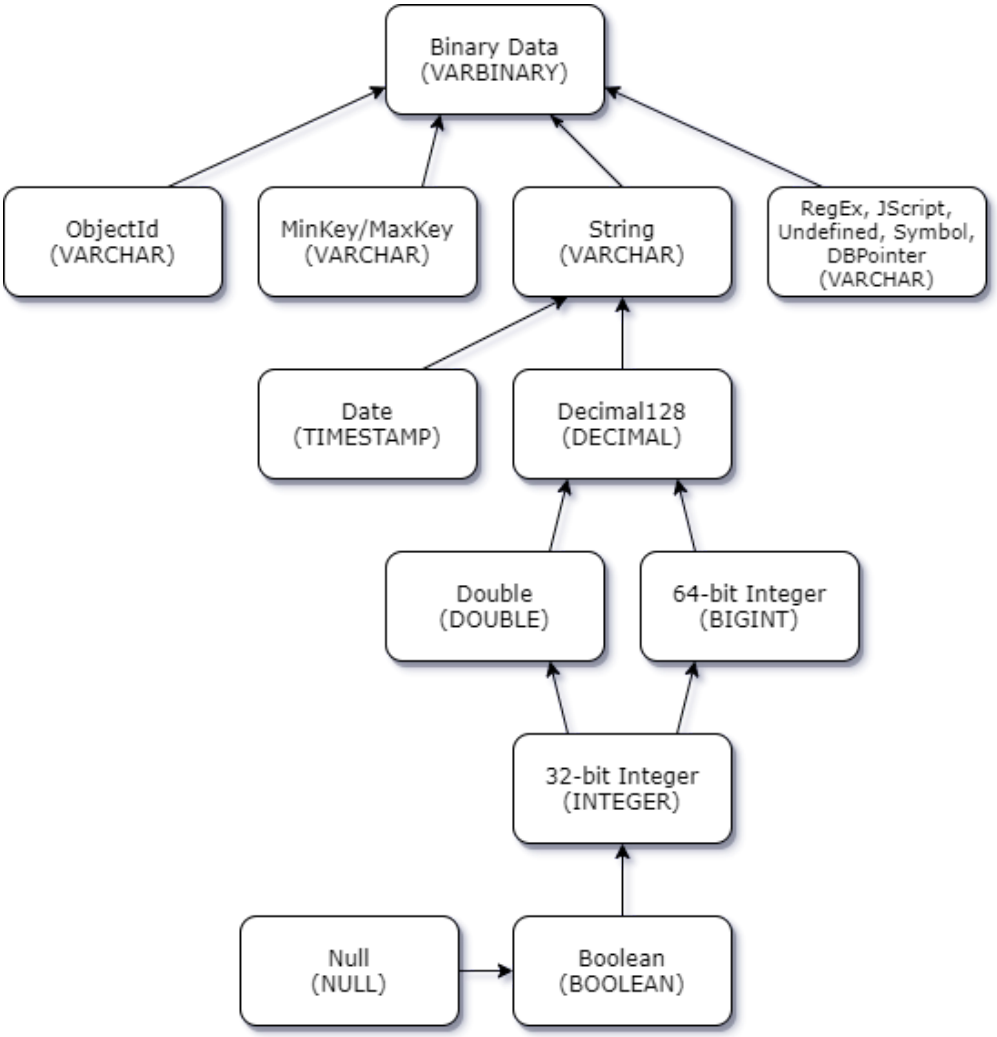
{
"_id" : "112244",
"name" : "Benjamin",
"age" : "32"
}
```

El campo age (edad) es de tipo entero de 32 bits en el primer documento, pero de tipo cadena en el segundo documento. En este caso, el controlador JDBC promoverá el tipo de datos JDBC a VARCHAR para gestionar cualquier tipo de datos cuando los encuentre.

Nombre de la tabla	Nombre de la columna	Tipo de datos	Key
equipo	id de equipo	VARCHAR	PK
equipo	name	VARCHAR	
equipo	edad	VARCHAR	

Scalar-scalar promoción de conflictos

El siguiente diagrama muestra la forma en que se resuelven los conflictos entre tipos de datos escalares-escalares.



Scalar-complex tipo promoción de conflictos

Al igual que ocurre con los conflictos entre tipos escalares-escalares, un mismo campo en diferentes documentos puede tener tipos de datos conflictivos entre complejos (matriz y objeto) y escalares (enteros, booleanos, etc.). Todos estos conflictos se resuelven (se transfieren) a VARCHAR para esos campos. En este caso, los datos de matriz y objeto se devuelven como representación JSON.

Ejemplo de conflicto entre matriz incrustada y campo de cadena:

```
{
   "_id":"112233",
   "name":"George Jackson",
   "subscriptions":[
      "Vogue",
      "People",
      "USA Today"
   ]
}
{
   "_id":"112244",
   "name":"Joan Starr",
   "subscriptions":1
}
```

El ejemplo anterior se corresponde con el esquema de la tabla customer2:

Nombre de la tabla	Nombre de la columna	Tipo de datos	Key
customer2	Id customer2	VARCHAR	PK
customer2	name	VARCHAR	
customer2	Suscripción	VARCHAR	

y la tabla virtual customer1_subscriptions:

Nombre de la tabla	Nombre de la columna	Tipo de datos	Key
customer1_subscriptions	Id customer1	VARCHAR	PK/FK
customer1_subscriptions	suscriptions_index_lvl0	BIGINT	PK
customer1_subscriptions	valor	VARCHAR	
customer_address	ciudad	VARCHAR	
customer_address	región	VARCHAR	
customer_address	país	VARCHAR	
customer_address	código	VARCHAR	

Manejo de tipos de datos de objetos y matrices

Hasta ahora, solo hemos descrito cómo se asignan los tipos de datos escalares. Los tipos de datos Object (objeto) y Array (matriz) están (actualmente) asignados a tablas virtuales. El controlador JDBC creará una tabla virtual para representar los campos de objeto o matriz de un documento. El nombre de la tabla virtual asignada concatenará el nombre de la colección original seguido del nombre del campo separado por un carácter de subrayado (“_”).

La clave principal de la tabla base (“_id”) adopta un nuevo nombre en la nueva tabla virtual y se proporciona como clave externa a la tabla base asociada.

En el caso de los campos de tipo matriz incrustados, las columnas de índice se generan para representar el índice de la matriz en cada nivel de la matriz.

Ejemplo de campo de objeto incrustado

En el caso de los campos de objetos de un documento, el controlador JDBC crea una asignación a una tabla virtual.

```
{
```

```
   "Collection: customer",
   "_id":"112233",
   "name":"George Jackson",
   "address":{
      "address1":"123 Avenue Way",
      "address2":"Apt. 5",
      "city":"Hollywood",
      "region":"California",
      "country":"USA",
      "code":"90210"
   }
}
```

El ejemplo anterior se asigna al esquema de la tabla de clientes:

Nombre de la tabla	Nombre de la columna	Tipo de datos	Key
cliente	Id del cliente	VARCHAR	PK
cliente	name	VARCHAR	

y la tabla virtual custome_address:

Nombre de la tabla	Nombre de la columna	Tipo de datos	Key
customer_address	Id del cliente	VARCHAR	PK/FK
customer_address	dirección1	VARCHAR	
customer_address	dirección2	VARCHAR	
customer_address	ciudad	VARCHAR	
customer_address	región	VARCHAR	
customer_address	país	VARCHAR	
customer_address	código	VARCHAR	

Ejemplo de campo de matriz incrustado

En el caso de los campos de objetos de un documento, el controlador JDBC crea una asignación a una tabla virtual.

```
{
   "Collection: customer1",
   "_id":"112233",
   "name":"George Jackson",
   "subscriptions":[
      "Vogue",
      "People",
      "USA Today"
   ]
}
```

El ejemplo anterior se asigna al esquema de la tabla customer1:

Nombre de la tabla	Nombre de la columna	Tipo de datos	Key
customer1	Id customer1	VARCHAR	PK
customer1	name	VARCHAR	

y la tabla virtual customer1_subscriptions:

Nombre de la tabla	Nombre de la columna	Tipo de datos	Key
customer1_subscriptions	Id customer1	VARCHAR	PK/FK
customer1_subscriptions	suscriptions_index_lvl0	BIGINT	PK
customer1_subscriptions	valor	VARCHAR	

Nombre de la tabla	Nombre de la columna	Tipo de datos	Key
customer_address	ciudad	VARCHAR	
customer_address	región	VARCHAR	
customer_address	país	VARCHAR	
customer_address	código	VARCHAR	

Compatibilidad y limitaciones de SQL

El controlador JDBC de Amazon DocumentDB es un controlador de solo lectura que admite un subconjunto de extensiones comunes y algunas de ellas. SQL-92 Consulte la documentación sobre las limitaciones de SQL y de JDBC para obtener más información.

Resolución de problemas

Si tiene problemas al utilizar el controlador JDBC de Amazon DocumentDB, consulte la Guía de solución de problemas.

Conexión mediante el controlador ODBC de Amazon DocumentDB

El controlador ODBC para Amazon DocumentDB proporciona una SQL-relational interfaz para desarrolladores y permite la conectividad desde herramientas de BI como Power BI Desktop y Microsoft Excel.

Para obtener información más detallada, consulte la documentación del controlador ODBC de Amazon DocumentDB sobre. GitHub

Temas

- Introducción
- Configuración del controlador ODBC de Amazon DocumentDB en Windows
- Conexión a Amazon DocumentDB desde Microsoft Excel
- Conexión a Amazon DocumentDB desde Power BI Desktop
- Generación automática de esquemas

- Compatibilidad y limitaciones de SQL
- Resolución de problemas

Introducción

Paso 1. Creación de clústeres de Amazon DocumentDB

Si todavía no tiene un clúster de Amazon DocumentDB, hay varias formas de empezar.

> **Note**
>
> Amazon DocumentDB es un servicio exclusivo de nube privada virtual (VPC). Si se conecta desde una máquina local fuera de la VPC del clúster, debe crear una conexión SSH a una instancia de Amazon EC2. Para obtener más información, consulte Conectarse mediante Amazon EC2. Para obtener más información sobre los túneles SSH, consulte Uso de un túnel SSH para conectarse a Amazon DocumentDB.

Paso 2. Instalación de JRE o JDK

En función de la aplicación de BI que utilice, puede que necesite instalar en su equipo una versión 8 o posterior de JRE o JDK de 64 bits. Puede descargar el Java SE Runtime Environment 8 aquí.

Paso 3. Descarga del controlador ODBC de Amazon DocumentDB

Descargue el controlador ODBC de Amazon DocumentDB aquí. Elija el instalador adecuado (por ejemplo, documentdb-odbc-1.0.0.msi). Siga la guía de instalación.

Paso 4. Cómo usar un túnel SSH para conectarse a Amazon DocumentDB

Los clústeres de Amazon DocumentDB se implementan dentro de una Amazon Virtual Private Cloud (Amazon VPC). Las instancias de Amazon EC2 u otros AWS servicios que estén implementados en la misma Amazon VPC pueden acceder a ellos directamente. Además, las instancias de Amazon EC2 u otros AWS servicios de diferentes VPC de la misma Región de AWS región o de otras pueden acceder a Amazon DocumentDB mediante la interconexión de VPC.

Sin embargo, supongamos que su caso de uso requiere que usted o su aplicación tengan acceso a los recursos de Amazon DocumentDB desde fuera de la VPC del clúster. Este será el caso de la mayoría de los usuarios que no ejecuten su aplicación en una máquina virtual de la misma VPC

que el clúster de Amazon DocumentDB. Si se conecta desde fuera de la VPC, puede utilizar la información sobre los túneles SSH (también conocida como reenvío de puertos) para acceder a los recursos de Amazon DocumentDB.

Para crear un túnel SSH, necesita una instancia de Amazon EC2 que se ejecute en la misma VPC de Amazon que el clúster de Amazon DocumentDB. Puede usar una instancia EC2 existente en la misma VPC que el clúster o crear una. Si es así, puede configurar un túnel de SSH en el clúster de Amazon DocumentDB `sample-cluster.node.us-east-1.docdb.amazonaws.com` ejecutando el siguiente comando en su equipo local:

```
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
```

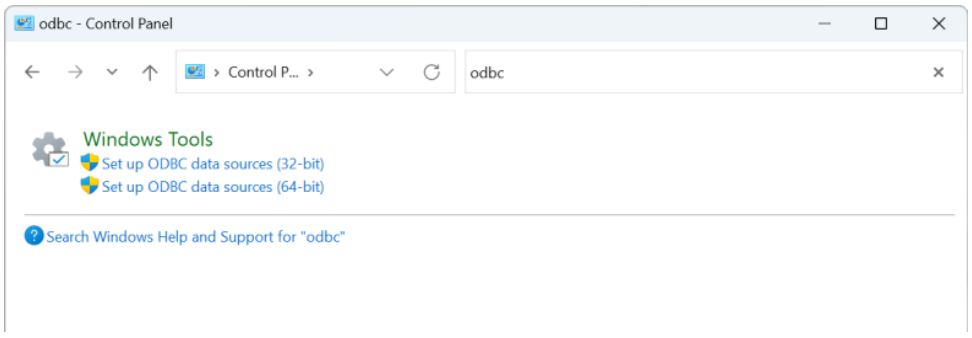
La marca `-L` se utiliza para el reenvío de un puerto local. Este es un requisito previo para conectarse a cualquier herramienta de BI que se ejecute en un cliente externo a su VPC. Una vez realizado el paso anterior, puede pasar a los siguientes con la herramienta de BI que prefiera.

Para obtener más información sobre los túneles SSH, consulte la documentación sobre el uso de un túnel SSH para conectarse a Amazon DocumentDB.

Configuración del controlador ODBC de Amazon DocumentDB en Windows

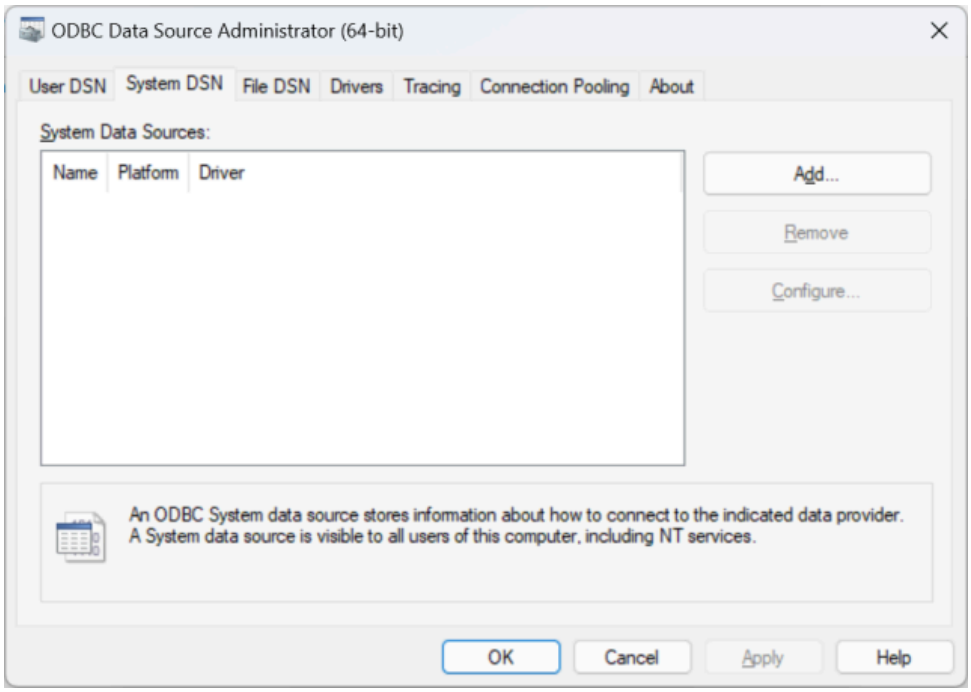
Utilice el siguiente procedimiento para configurar el controlador ODBC de Amazon DocumentDB en Windows:

1. Abra el Panel de control en Windows y busque ODBC (o, en el menú, seleccione Herramientas de Windows > Fuentes de datos ODBC (32 bits) o Fuentes de datos ODBC (64 bits)):

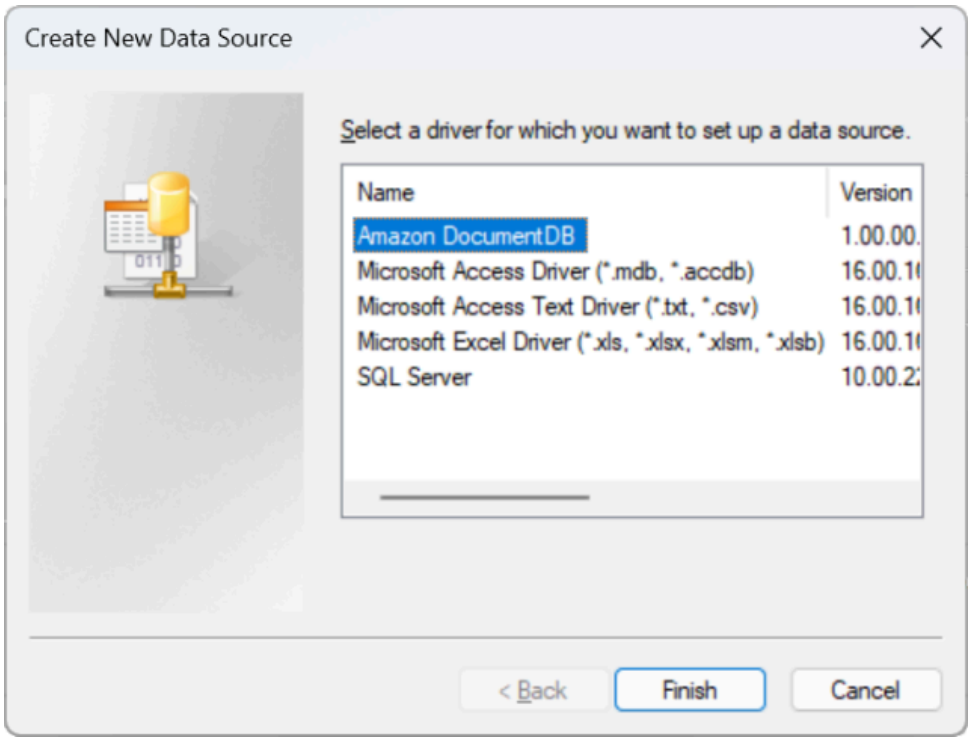


2. Seleccione el administrador de origen de datos del controlador ODBC adecuado: elija la versión de 32 bits si está instalada; de lo contrario, elija la versión de 64 bits.

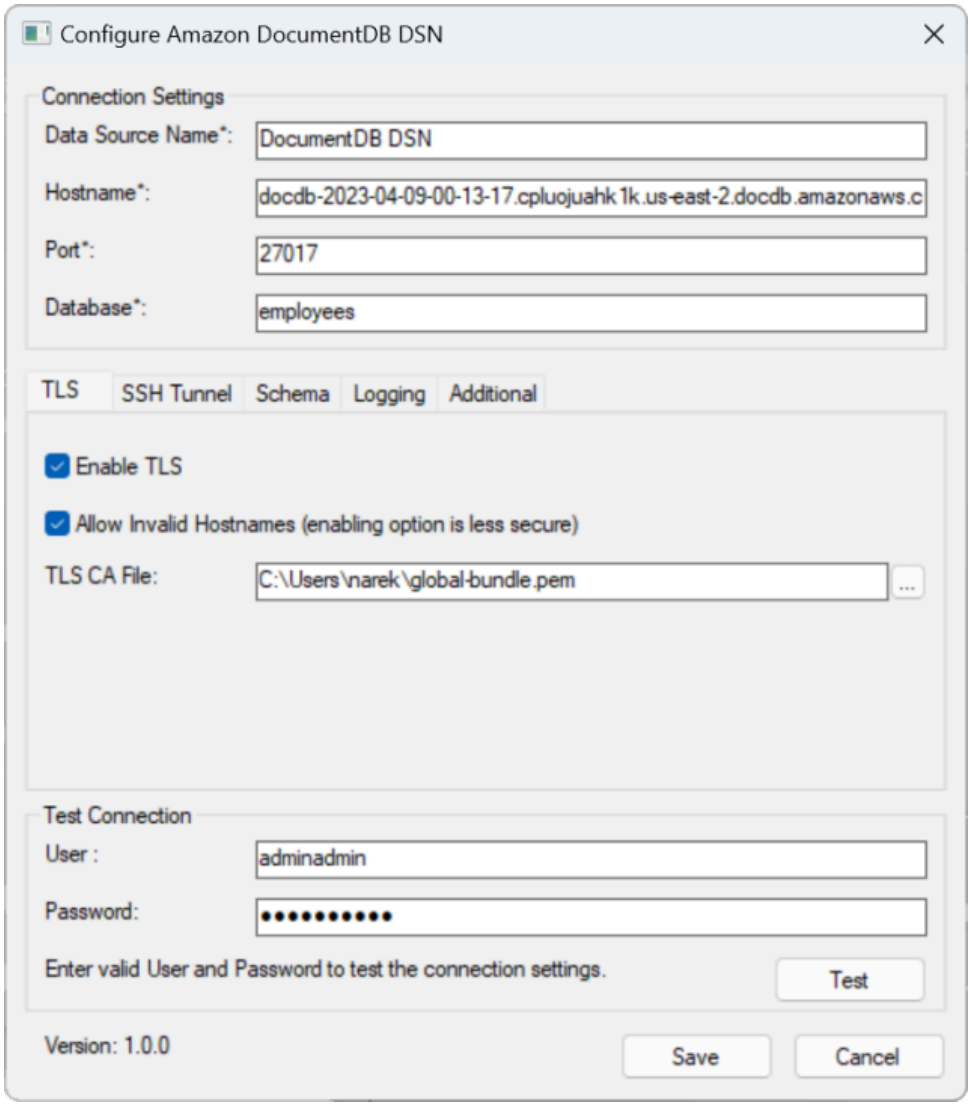
3. Seleccione la pestaña DSN del sistema y, a continuación, elija Agregar... para agregar un DSN nuevo:



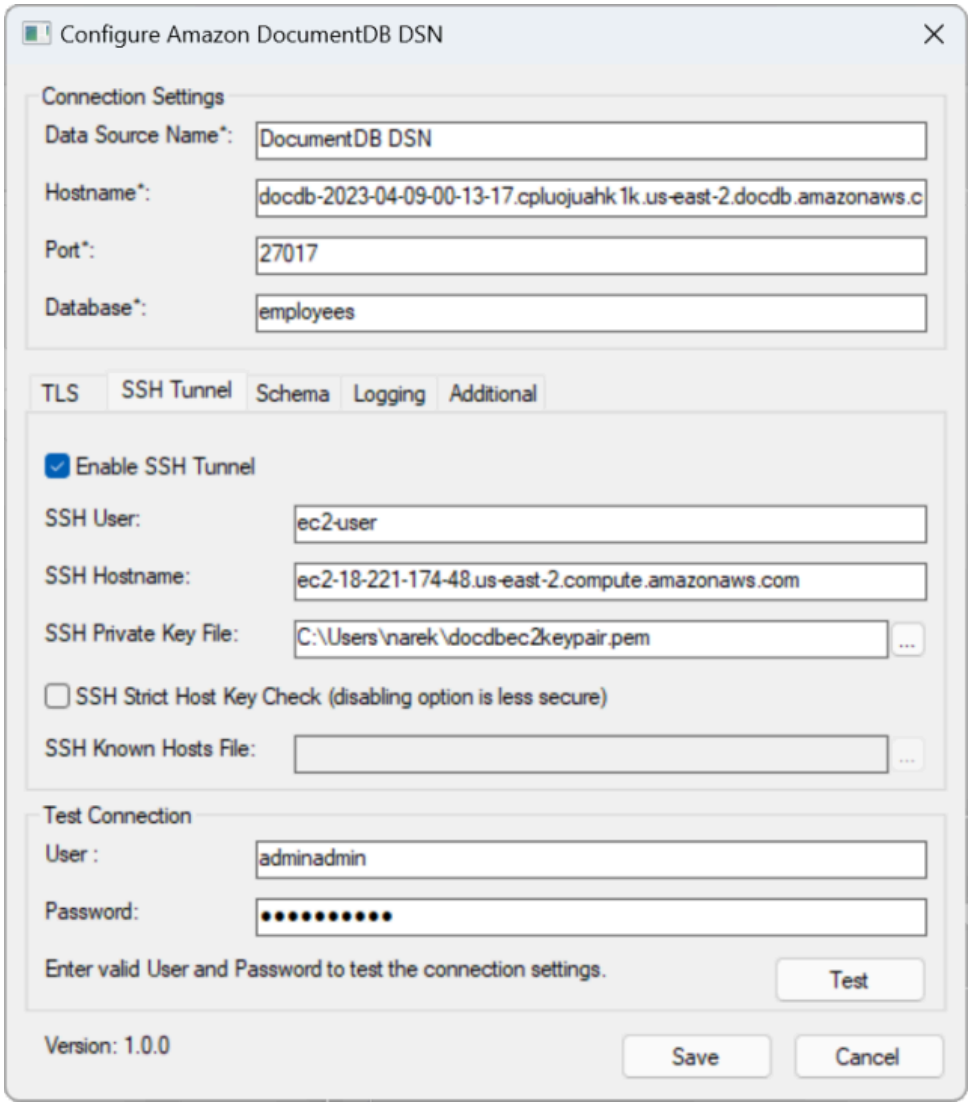
4. Elija Amazon DocumentDB en la lista de controladores de origen de datos:



5. En el cuadro de diálogo Configurar el DSN de Amazon DocumentDB, complete los campos Configuración de la conexión, la pestaña TLS y Probar la conexión y, a continuación, seleccione Guardar:



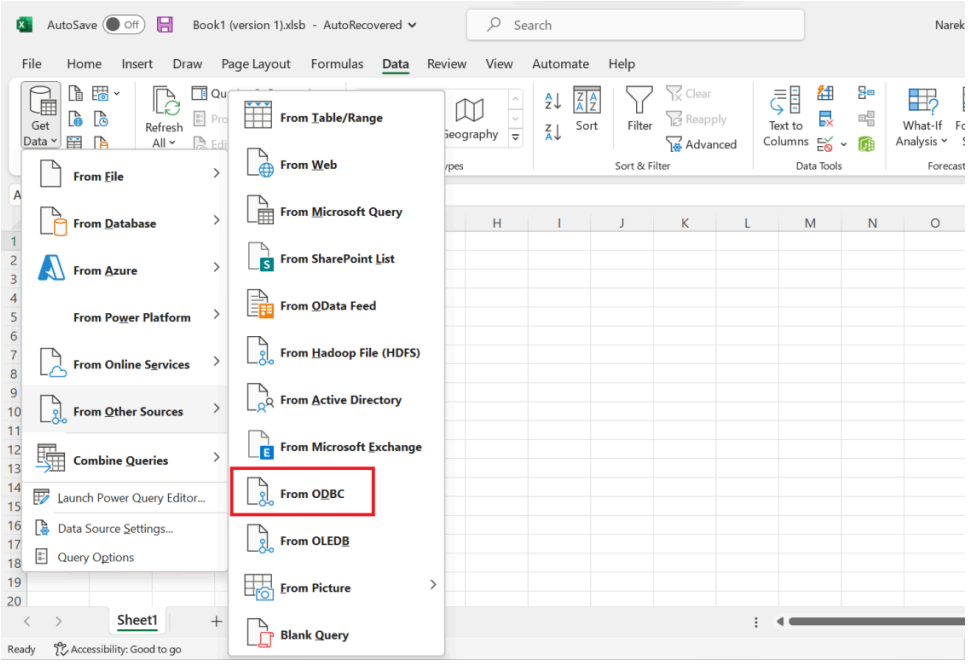
6. Asegúrese de completar el formulario de Windows correctamente, ya que los detalles de la conexión variarán según el método de tunelización SSH que haya elegido para la instancia EC2. Consulte los métodos de tunelización SSH aquí. Consulte Sintaxis y opciones de la cadena de conexión para obtener más información sobre cada propiedad.



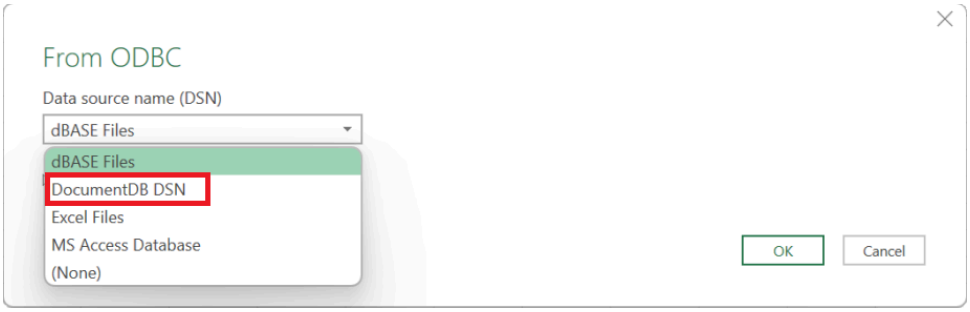
Para obtener más información sobre la configuración del controlador ODBC de Amazon DocumentDB en Windows, consulte aquí. https://github.com/aws/amazon-documentdb-odbc-driver/blob/develop/src/markdown/setup/windows-dsn-configuration.md

Conexión a Amazon DocumentDB desde Microsoft Excel

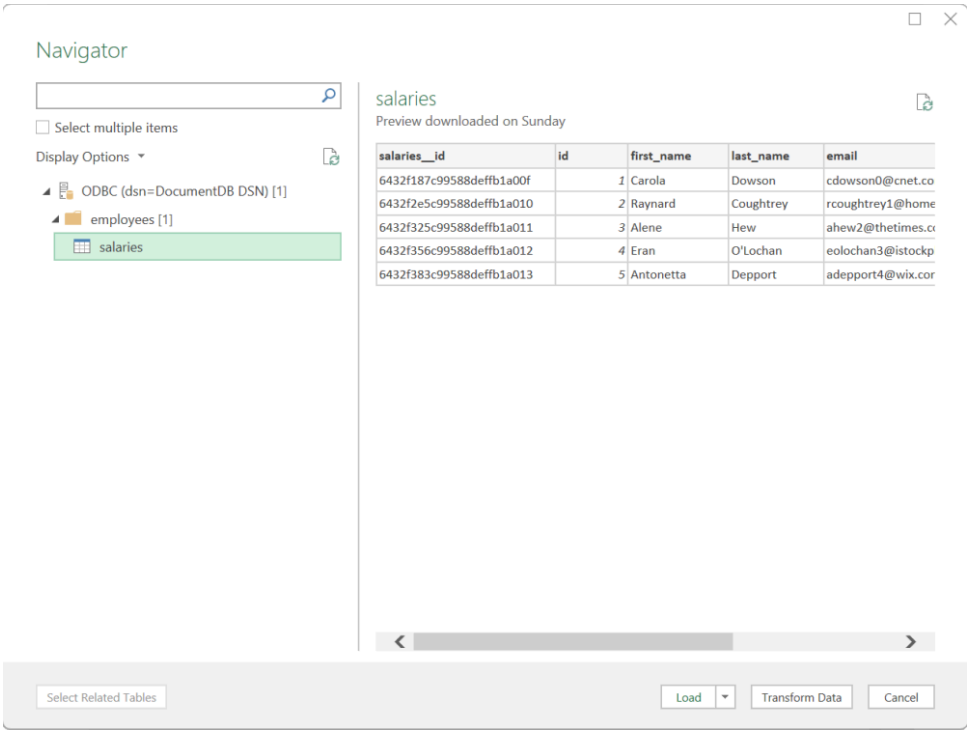
1. Asegúrese de que el controlador Amazon DocumentDB se haya instalado y configurado correctamente. Para obtener más información, consulte Configuración del controlador ODBC de Amazon DocumentDB en Windows.
2. Abra Microsoft Excel.
3. Navegue hasta Datos > Obtener datos > De otras fuentes.
4. Elija Desde ODBC:



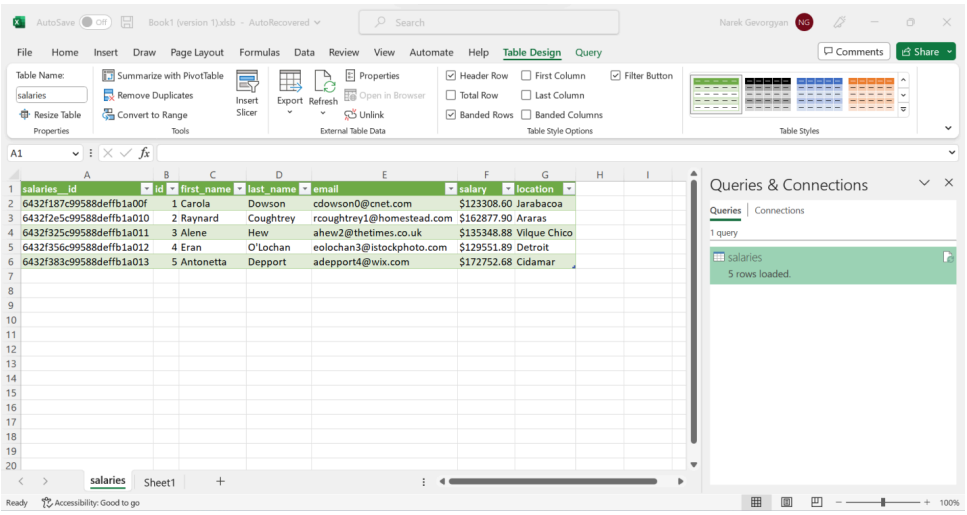
5. Seleccione el origen de datos en el menú desplegable Nombre del origen de datos (DSN) asociado a Amazon DocumentDB:



6. Elija la colección desde la que desea cargar datos en Excel:



7. Cargar datos en Excel:



Conexión a Amazon DocumentDB desde Power BI Desktop

Temas

- Requisitos previos
- Cómo añadir un conector personalizado de Microsoft Power BI Desktop
- Conexión mediante el conector personalizado de Amazon DocumentDB
- Configuración de la puerta de enlace Power BI de Microsoft

Requisitos previos

Antes de empezar, asegúrese de que el controlador ODBC de Amazon DocumentDB esté instalado correctamente.

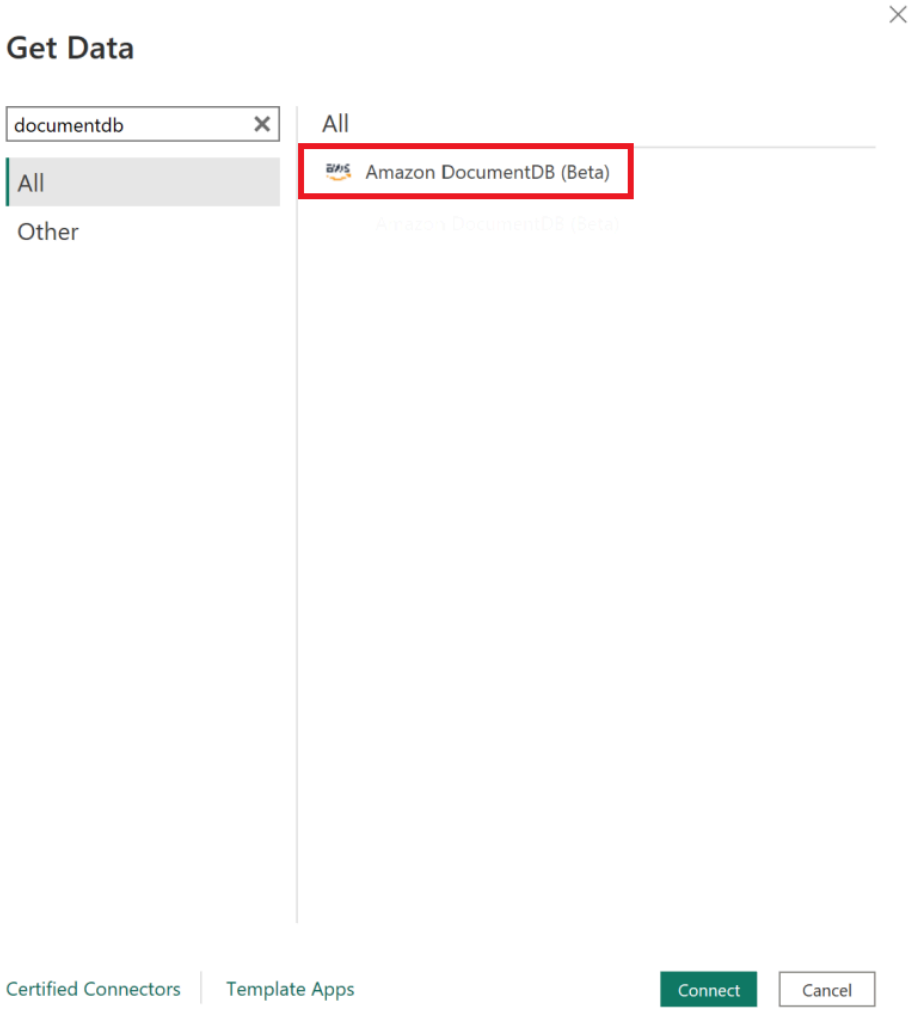
Cómo añadir un conector personalizado de Microsoft Power BI Desktop

Copie el `AmazonDocumentDBConnector.mez` archivo a la `<User>\Documents\Power BI Desktop\Custom Connectors\` carpeta (o a, `<User>\OneDrive\Documents\Power BI Desktop\Custom Connectors` si lo usa OneDrive). Esto permitirá a Power BI acceder al conector personalizado. Puede obtener el conector a Power BI Desktop aquí. Reinicie Power BI Desktop para asegurarse de que el conector esté cargado.

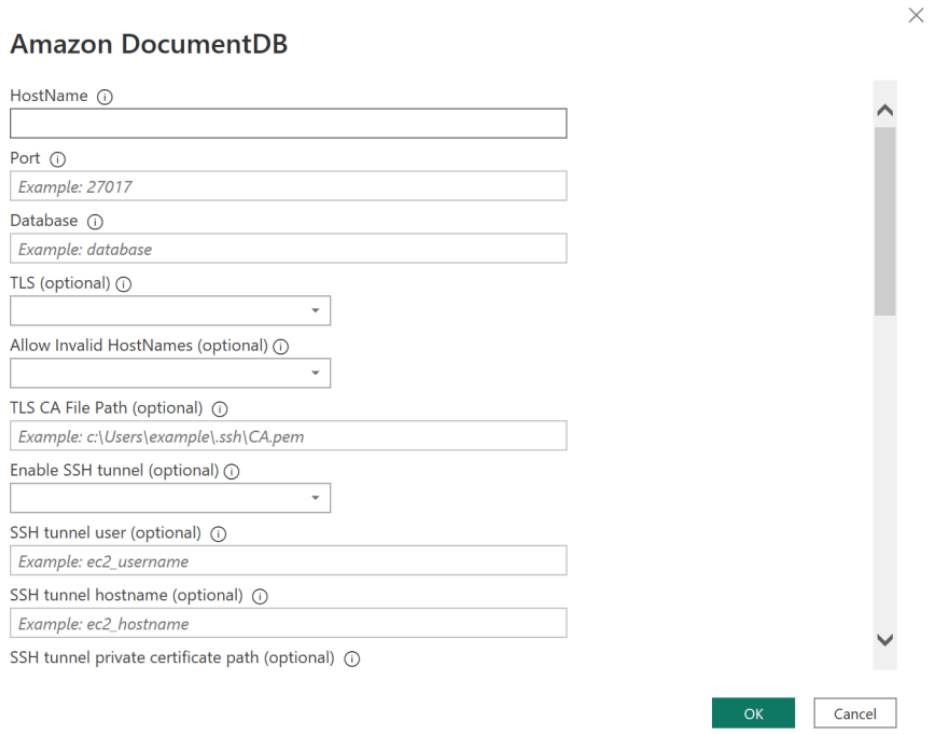
> **Note**
>
> El conector personalizado solo admite el nombre de usuario y la contraseña de Amazon DocumentDB para la autenticación.

Conexión mediante el conector personalizado de Amazon DocumentDB

1. Seleccione Amazon DocumentDB (Beta) en Obtener datos y seleccione Conectar. Si recibe una advertencia por el uso de un servicio de terceros, elija Continuar.



2. Introduzca toda la información necesaria para conectarse a su clúster de Amazon DocumentDB y, a continuación, seleccione Aceptar:



> **Note**
>
> Según la configuración del nombre del origen de datos (DSN) del controlador ODBC, es posible que la pantalla de detalles de la conexión SSH no aparezca si ya ha proporcionado la información necesaria en la configuración del DSN.

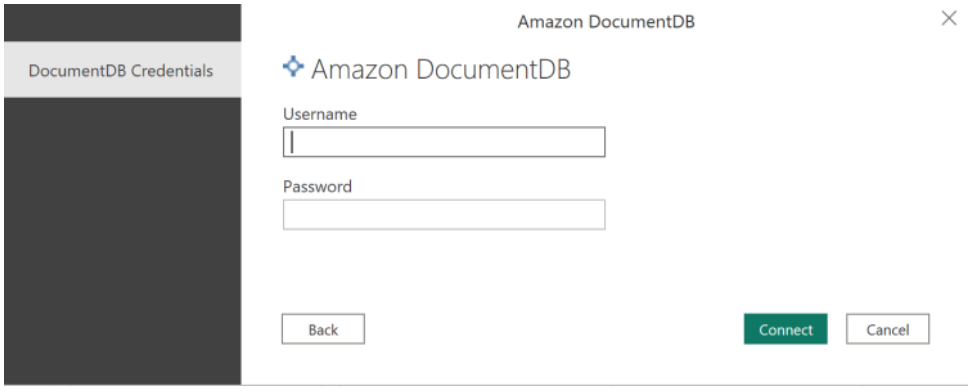
3. Seleccione el modo de conectividad de datos:

   - Importación: carga todos los datos y almacena la información en el disco. Los datos deben actualizarse y volver a cargarse para mostrar las actualizaciones de datos.
   - Consulta directa: no carga datos, pero realiza consultas en tiempo real sobre los datos. Esto significa que no es necesario actualizar y volver a cargar los datos para mostrar las actualizaciones de datos.

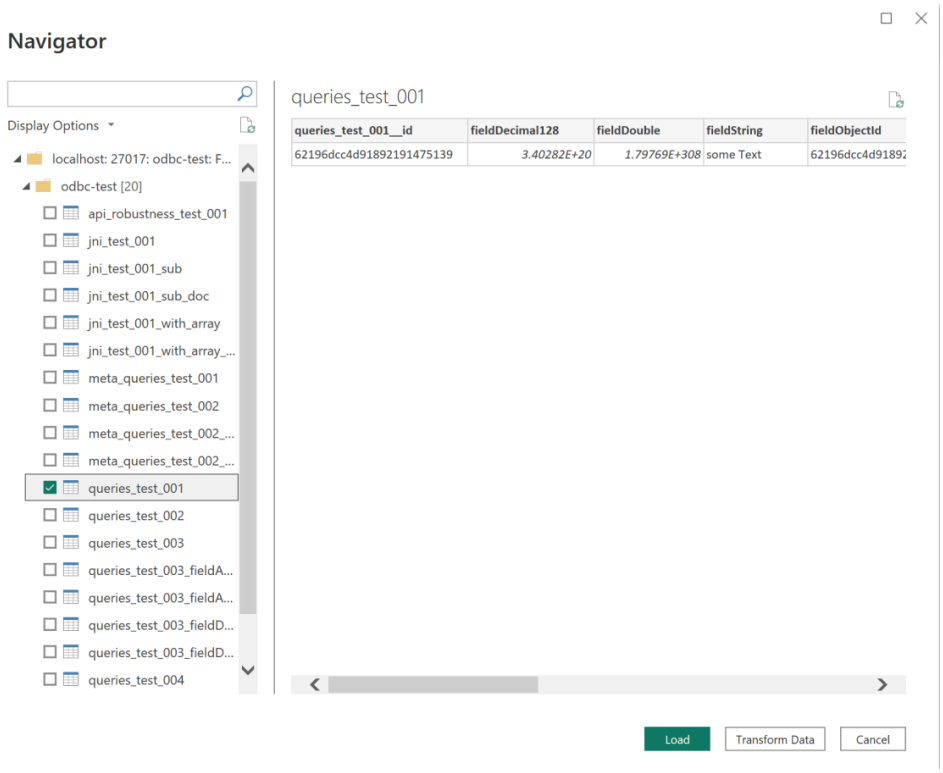


> **Note**
>
> Si utiliza un conjunto de datos muy grande, es posible que la importación de todos los datos tarde más tiempo.

4. Si es la primera vez que se conecta a este origen de datos, seleccione el tipo de autenticación e introduzca sus credenciales cuando así se le pida. A continuación, elige Conectar:



5. En el cuadro de diálogo del Navegador, seleccione las tablas de base de datos que desee y, a continuación, elija Cargar para cargar los datos o Transformar datos para continuar con la transformación de los datos.



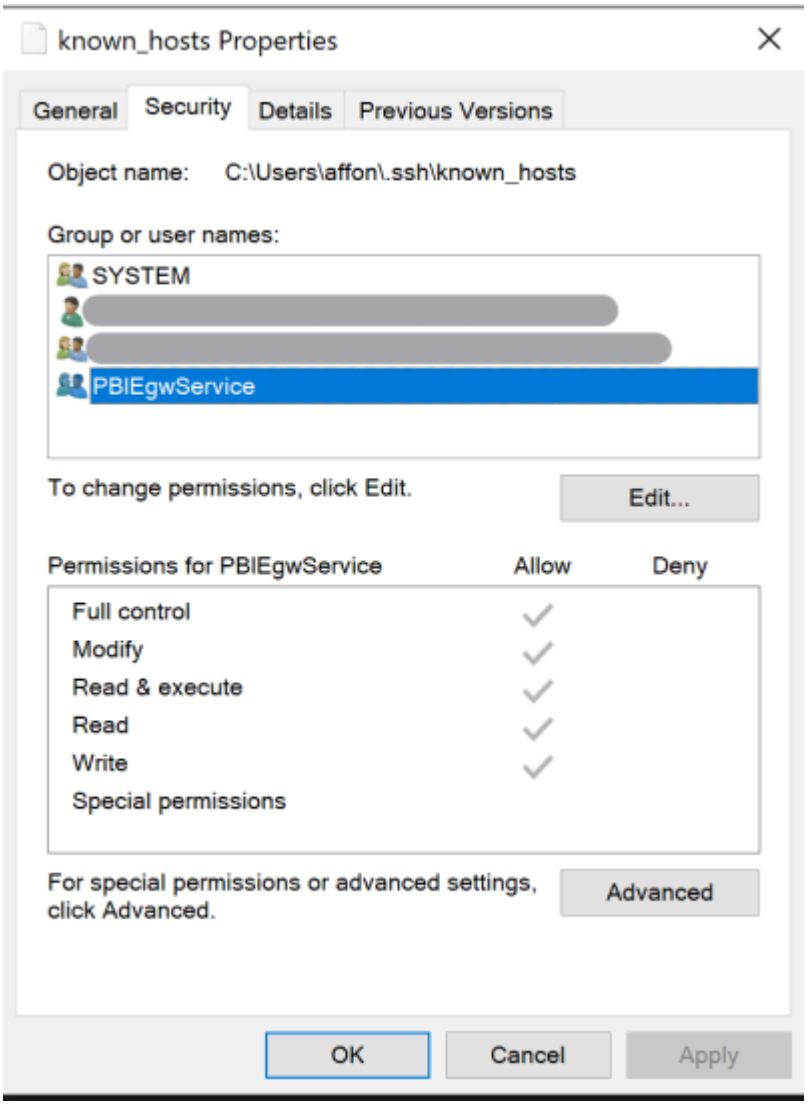
> **Note**
>
> La configuración del origen de datos se guarda una vez que se conecta. Para modificarlos, seleccione Transformar datos > Configuración del origen de datos.

Configuración de la puerta de enlace Power BI de Microsoft

Requisitos previos:

- Asegúrese de que el conector personalizado funcione con la puerta de enlace Power BI.
- Asegúrese de que el DSN de ODBC esté creado en el origen de datos de ODBC en la pestaña Sistema de la máquina en la que está instalada la puerta de enlace Power BI.

Si utiliza la característica de túnel SSH interno, el archivo `known_hosts` debe estar ubicado en un lugar donde la cuenta de servicio de Power BI tenga acceso a él.



> **Note**
>
> Esto también se aplica a cualquier archivo que necesite para poder establecer una conexión con su clúster de Amazon DocumentDB, como un archivo de certificado (archivo pem) de una autoridad de certificación (CA).

Generación automática de esquemas

El controlador ODBC utiliza el controlador JDBC de Amazon DocumentDB a través de JNI (interfaz nativa de Java), lo que hace que la característica de generación automática de esquemas funcione de manera similar en el controlador JDBC. Para obtener más información sobre la generación automática de esquemas, consulte. Generación automática de esquemas JDBC Para obtener más

información sobre la arquitectura de controladores ODBC, consulte la arquitectura de controladores ODBC de Amazon DocumentDB en. GitHub

Compatibilidad y limitaciones de SQL

El controlador ODBC de Amazon DocumentDB es un controlador de solo lectura que admite un subconjunto de extensiones comunes y algunas de ellas. SQL-92 Consulte la documentación de Compatibilidad y limitaciones de ODBC para obtener más información.

Resolución de problemas

Si tiene problemas al utilizar el controlador ODBC de Amazon DocumentDB, consulte la Guía de solución de problemas.

Programación con Amazon DocumentDB

El servicio admite la validación de esquemas JSON.

Temas

- Guía de programación de Amazon DocumentDB en Java
- Cómo utilizar la validación de esquemas JSON

Guía de programación de Amazon DocumentDB en Java

Esta guía completa proporciona una guía detallada para trabajar con Amazon DocumentDB mediante los controladores Java de MongoDB y abarca aspectos esenciales de las operaciones y la administración de bases de datos.

Temas

- Introducción
- Requisitos previos
- Modelos de datos para esta guía
- Conexión a Amazon DocumentDB con un controlador Java de MongoDB
- Realización de operaciones CRUD en Amazon DocumentDB con Java
- Administración de índices en Amazon DocumentDB con Java

- Event-driven programación con Amazon DocumentDB y Java

Introducción

La guía comienza con la conectividad y explica cómo establecer conexiones seguras a los clústeres de DocumentDB mediante el controlador Java de MongoDB. En él se detallan los componentes de la cadena de conexión, SSL/TLS la implementación y las diversas opciones de conexión, incluida la autenticación de IAM y la agrupación de conexiones, junto con estrategias sólidas de gestión de errores.

En la sección de operaciones CRUD (crear, leer, actualizar, eliminar) de la guía, se trata exhaustivamente la manipulación de documentos y muestra cómo crear, leer, actualizar y eliminar documentos mediante operaciones únicas o masivas. Se explica el uso de filtros, consultas y diversas opciones de operación, al tiempo que se hace hincapié en las mejores prácticas para la gestión de errores y la implementación de la lógica de reintento para mejorar la fiabilidad. La guía también cubre ampliamente la gestión de índices y detalla la creación y el mantenimiento de diferentes tipos de índices, incluidos los índices de un solo campo, compuestos, dispersos y de texto. Se explica cómo optimizar el rendimiento de las consultas mediante una selección adecuada de los índices y el uso de la función `explain()` para analizar los planes de ejecución de consultas.

La última sección se centra en la programación basada en eventos mediante los flujos de cambios de Amazon DocumentDB y demuestra cómo implementar la supervisión de cambios de datos en tiempo real en aplicaciones Java. Abarca la implementación de cursores de flujo de cambios, la gestión de los tokens de reanudación para un funcionamiento continuo y las operaciones basadas en el tiempo para el procesamiento de datos históricos. A lo largo de la guía, se proporcionan ejemplos de código prácticos y prácticas recomendadas, lo que la convierte en un recurso inestimable para crear aplicaciones sólidas en Java con Amazon DocumentDB.

Requisitos previos

Antes de empezar, asegúrese de que tiene lo siguiente:

- Y Cuenta de AWS con un clúster de Amazon DocumentDB configurado. Para ver la configuración del clúster, consulte Cómo empezar a usar Amazon DocumentDB con Amazon EC2 en el AWS blog de bases de datos.
- El kit de desarrollo de Java (JDK) está instalado (utilizaremos Amazon Corretto 21 para esta guía).
- Maven para la administración de dependencias.

Modelos de datos para esta guía

Todo el código de ejemplo de esta guía presupone una conexión a una base de datos de prueba «ProgGuideData» que tiene una colección «Restaurantes». Todos los códigos de muestra de esta guía funcionan en un sistema de listas de restaurantes y, a continuación, se muestra un ejemplo del aspecto de un documento de este 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
    }
}
```

Todos los ejemplos de código en los que se muestran las operaciones CRUD, la gestión de índices y la programación basada en eventos suponen que tiene un `dbClient` objeto de `MongoClient`, una `connectionDB` objeto de `MongoDatabase` y una `collection` objeto de `MongoCollection`.

> **Note**
>
> Todos los ejemplos de código de esta guía se han probado con el controlador Java de MongoDB, versión 5.3.0.

Conexión a Amazon DocumentDB con un controlador Java de MongoDB

En esta sección se proporciona una guía paso a paso para conectarse a Amazon DocumentDB mediante controladores Java. Esto le permitirá empezar a integrar DocumentDB en sus aplicaciones Java.

Temas

- Paso 1: Configuración del proyecto
- Paso 2: crear una cadena de conexión
- Paso 3: escribir el código de conexión
- Paso 4: gestionar excepciones de conexión
- Paso 5: ejecutar el código
- Prácticas recomendadas de conexión

Paso 1: Configuración del proyecto

1. Cree un proyecto java con Maven:

   ```
   mvn archetype:generate -DgroupId=com.docdb.guide -DartifactId=my-docdb-project -
   DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
   ```

2. Añada el controlador Java de MongoDB como una dependencia para el proyecto en su archivo «pom.xml»:

```
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>5.3.0</version>
</dependency>
```

Paso 2: crear una cadena de conexión

La cadena de conexión de Amazon DocumentDB es esencial para establecer una conexión entre la aplicación y el clúster de DocumentDB. Esta cadena encapsula información crucial, como el punto de conexión del clúster, el puerto, los detalles de autenticación y varias opciones de conexión. Para crear una cadena de conexión de DocumentDB, normalmente se empieza con el formato básico:

```
"mongodb://username:password@cluster-endpoint:port/?[connection options]"
```

Deberá sustituir los términos «nombre de usuario» y «contraseña» por sus credenciales reales. Puede encontrar el punto final y el número de puerto de su clúster en el Consola de administración de AWS y a través del AWS CLI. Consulte Búsqueda de puntos de conexión de un clúster para encontrar el punto de conexión del clúster. El puerto 27017 es el puerto predeterminado de DocumentDB.

Ejemplos de cadenas de conexión

- Para establecer una conexión a DocumentDB mediante cifrado en tránsito y asegurarse de que las solicitudes de lectura vayan a las réplicas de lectura y escriban a la principal:

```
"mongodb://username:password@cluster-endpoint:27017/?tls=true&
   tlsCAFile=global-bundle.pem&
   readPreference=secondaryPreferred&
   retryWrites=false"
```

- Para establecer una conexión a DocumentDB mediante la autenticación de IAM:

```
"mongodb://cluster-endpoint:27017/?tls=true&
   tlsCAFile=global-bundle.pem&
   readPreference=secondaryPreferred&
   retryWrites=false&
   authSource=%24external&
   authMechanism=MONGODB-AWS"
```

Las diferentes opciones disponibles para la cadena de conexión son las siguientes:

- Certificado TLS
- Lectura a partir de réplicas de lectura
- Write concern y registro en diario
- RetryWrites
- Autenticación de IAM
- Grupo de conexiones
- Parámetros de tiempo de espera de conexión

Certificado TLS

`tls=true|false`: esta opción habilita o deshabilita la seguridad de la capa de transporte (TLS). De forma predeterminada, el cifrado en tránsito está habilitado en el clúster de Amazon DocumentDB y, por lo tanto, a menos que TLS esté deshabilitada en el nivel del clúster, el valor de esta opción debería ser `true`.

Cuando se utiliza TLS, el código debe proporcionar un certificado SSL cuando crea la conexión a un clúster de DocumentDB. Descargue el certificado necesario para realizar la conexión segura al clúster: `global-bundle.pem`. Hay dos formas de usar el archivo `global-bundle.pem`.

- Opción 1: extraer todos los certificados del archivo `global-bundle.pem` y usar la herramienta clave de Java para almacenarlos en un archivo `.jks` que luego se pueda usar en el código. Consulte la pestaña Java en Conexión con TLS habilitado para ver el script que muestra cómo hacerlo.
- Opción 2: añadir dinámicamente el archivo `global-bundle.pem` al código, crear un almacén de claves en memoria y usar `SSLContext` para proporcionar el certificado como parte de la conexión.

Lectura a partir de réplicas de lectura

`replicaSet=rs0&readPreference=secondaryPreferred`: si se especifican estas dos opciones, se redirigen todas las solicitudes de lectura a las réplicas de lectura y las solicitudes de escritura a la instancia principal. El uso de `replicaSet=rs0` en la cadena de conexión permite que el controlador MongoDB mantenga una vista actualizada automáticamente de la topología del clúster, lo que permite que las aplicaciones mantengan la visibilidad de las configuraciones de nodos actuales a medida que se añaden o eliminan instancias. Si no se proporcionan estas opciones ni

se especifica `readPreference=primary`, se envían todas las lecturas y escrituras a la instancia principal. Para obtener más opciones de `readPreference`, consulte Opciones de preferencia de lectura.

Write concern y registro en diario

La opción Write concern determina el nivel de reconocimiento solicitado a la base de datos para las operaciones de escritura. Los controladores MongoDB ofrecen una opción para ajustar los archivos de Write concern y de diario. Amazon DocumentDB no espera que establezca la opción de Write concern y diario, e ignora los valores enviados para `w` y `j` (`writeConcern` y `journal`). DocumentDB siempre escribe datos con `writeConcern`: `majority` y`journal`: `true` de modo que las escrituras se registren de forma duradera en la gran mayoría de los nodos antes de enviar una confirmación al cliente.

RetryWrites

`retryWrites`— A partir de la versión 8.0.2 del motor, Amazon DocumentDB admite escrituras que se pueden volver a intentar. Configúrelo para habilitarlo. `retryWrites=true` En las versiones del motor anteriores a la 8.0.2, defina `false` este atributo en. Para obtener más información, consulte Escrituras que se pueden volver a intentar en Amazon DocumentDB.

Autenticación de IAM

`authSource=%24external`y **`authMechanism=MONGODB-AWS`**: estos dos parámetros se utilizan para autenticar el uso. AWS Identity and Access Management En la actualidad, la autenticación de IAM está disponible solamente en la versión 5.0 del clúster basado en instancias. Para obtener más información, consulte Autenticación mediante la identidad de IAM.

Grupo de conexiones

Estas opciones están disponibles para la agrupación de conexiones:

- **`maxPoolSize`**: establece el número máximo de conexiones que se pueden crear en el grupo. Cuando todas las conexiones están en uso y llega una nueva solicitud, espera a que haya una conexión disponible. El valor predeterminado para los controladores Java de MongoDB es 100.
- **`minPoolSize`**: indica el número mínimo de conexiones que deben mantenerse en el grupo en todo momento. El valor predeterminado para los controladores Java de MongoDB es 0.
- **`maxIdleTimeMS`**: determina cuánto tiempo puede permanecer inactiva una conexión en el grupo antes de cerrarla y eliminarla. El valor predeterminado para los controladores Java de MongoDB es 100 milisegundos.

- **`waitQueueTimeoutMS`**: configura cuánto tiempo debe esperar un subproceso para que una conexión esté disponible cuando el grupo esté en su tamaño máximo. Si una conexión no está disponible en este tiempo, se produce una excepción. El valor predeterminado para los controladores Java de MongoDB es 120 000 milisegundos (2 minutos).

Parámetros de tiempo de espera de conexión

El tiempo de espera es un mecanismo para limitar el tiempo que puede tardar una operación o un intento de conexión antes de que se considere fallido. Los siguientes parámetros de tiempo de espera están disponibles para evitar esperas indefinidas y administrar la asignación de recursos:

- **`connectTimeoutMS`**: configura cuánto tiempo esperará el controlador para establecer una conexión con el clúster. El valor predeterminado es de 10 000 milisegundos (10 segundos).
- **`socketTimeoutMS`**: especifica cuánto tiempo esperará el controlador para recibir una respuesta del servidor para una operación que no sea de escritura. El valor predeterminado es 0 (sin tiempo de espera o es infinito).
- **`serverSelectionTimeoutMS`**: especifica cuánto tiempo esperará el controlador para encontrar un servidor disponible en el clúster. El valor predeterminado de esta configuración es de 30 segundos y es suficiente para que se elija una nueva instancia principal durante la conmutación por error.

Paso 3: escribir el código de conexión

En el siguiente ejemplo de código se muestra cómo establecer una conexión de TLS a Amazon DocumentDB:

- Crea los objetos `KeyStore` y `SSLContext>` de Java.
- También crea el objeto `MongoClientSettings` pasándolo al objeto `ConnectionString`. Para establecer una conexión TLS, debe usar el objeto `MongoClientSettings` para vincular la `connectionstring` y el `sslcontext`.
- El uso de `MongoClients` obtiene un objeto `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);
    }
}
```

Paso 4: gestionar excepciones de conexión

Cuando se trabaja con DocumentDB en aplicaciones Java, el manejo de las excepciones de conexión es crucial para mantener operaciones de base de datos sólidas y fiables. Si se gestionan adecuadamente, estas excepciones no solo ayudan a diagnosticar los problemas rápidamente, sino que también garantizan que la aplicación pueda gestionar sin problemas las interrupciones temporales de la red o la falta de disponibilidad del servidor, lo que mejora la estabilidad y

la experiencia del usuario. Algunas de las excepciones más importantes relacionadas con el establecimiento de la conexión son las siguientes:

- **`MongoException`**: es una excepción general y podría emitirse en varias situaciones no cubiertas por excepciones más específicas. Asegúrese de que esta excepción se gestione después de todas las demás excepciones específicas, ya que se trata de una excepción general de MongoDB.
- **`MongoTimeoutException`**: se emite cuando se agota el tiempo de espera de una operación. Por ejemplo, cuando consulta un punto de conexión de un clúster que no existe.
- **`MongoSocketException`**: se emite por problemas relacionados con la red. Por ejemplo, una desconexión repentina de la red durante una operación.
- **`MongoSecurityException`**: se emite cuando se produce un error en la autenticación. Por ejemplo, cuando se intenta conectar con credenciales incorrectas.
- **`MongoConfigurationException`**: se emite cuando hay un error en la configuración del cliente. Por ejemplo, si se utiliza una cadena de conexión no válida.

Paso 5: ejecutar el código

El siguiente ejemplo de código crea una conexión a Amazon DocumentDB e imprime todas las bases de datos:

```
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);
    }
}
```

Prácticas recomendadas de conexión

Las siguientes son prácticas recomendadas que se deben tener en cuenta cuando se conecta a Amazon DocumentDB con un controlador Java de MongoDB:

- Cierre siempre el `MongoClient` cuando ya no necesite que el cliente libere recursos.
- Gestione las excepciones de forma adecuada e implemente un registro de errores adecuado.

- Utilice variables de entorno o AWS Secrets Manager para almacenar información confidencial, como nombres de usuario y contraseñas.

Realización de operaciones CRUD en Amazon DocumentDB con Java

En esta sección se describe cómo realizar las operaciones CRUD (crear, leer, actualizar, eliminar) en Amazon DocumentDB con los controladores Java de MongoDB.

Temas

- Creación e inserción de documentos en una colección de DocumentDB
- Lectura y recuperación de datos de una colección de DocumentDB
- Actualización de documentos existentes en una colección de DocumentDB
- Eliminación de documentos de una colección de DocumentDB
- Gestión de errores con lógica de reintento

Creación e inserción de documentos en una colección de DocumentDB

La inserción de documentos en Amazon DocumentDB le permite añadir nuevos datos a sus colecciones. Existen varias formas de realizar las inserciones, en función de sus necesidades y del volumen de datos con el que esté trabajando. El método más básico para insertar un documento individual en la colección es `insertOne()`. Para insertar varios documentos a la vez, puede utilizar el método de `insertMany()`, que le permite añadir una serie de documentos en una sola operación. Otro método para insertar muchos documentos en una colección de DocumentDB es `bulkWrite()`. En esta guía, analizamos todos estos métodos para crear documentos en una colección de DocumentDB.

`insertOne()`

Empecemos por examinar cómo insertar un documento individual en una colección de Amazon DocumentDB. La inserción de un único documento se realiza mediante el método `insertOne()`. Este método toma una `BsonDocument` para la inserción y devuelve un `InsertOneResult` objeto que se puede utilizar para obtener el identificador de objeto del nuevo documento insertado. El siguiente código de ejemplo muestra la inserción de un documento de restaurante en la colección:

```
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;
}
```

Cuando use `insertOne()`, asegúrese de incluir una gestión de errores adecuada. Por ejemplo, en el código anterior, «`restaurantId`» tiene un índice único y, por lo tanto, cuando vuelva a ejecutar este código, se generará la siguiente `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()

Los métodos principales utilizados para insertar muchos documentos en una colección son insertMany() y `bulkWrite()`.

El método `insertMany()` es la forma más sencilla de insertar varios documentos en una sola operación. Acepta una lista de documentos y los inserta en la colección. Este método es ideal cuando se inserta un lote de documentos nuevos que son independientes entre sí y no requieren ningún procesamiento especial ni operaciones mixtas. El siguiente código muestra la lectura de documentos JSON de un archivo y su inserción en la colección. La función `insertMany()` devuelve un objeto `InsertManyResult`InsertManyResult que se puede utilizar para obtener los identificadores de todos los documentos insertados.

```
// 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()

El método `bulkWrite()` permite realizar múltiples operaciones de escritura (insertar, actualizar, eliminar) en un solo lote. Puede usar `bulkWrite()` cuando necesite realizar diferentes tipos de operaciones en un solo lote, como insertar algunos documentos y actualizar otros. `bulkWrite()` admite dos tipos de escritura por lotes, ordenada y desordenada:

- Operaciones ordenadas (predeterminado): Amazon DocumentDB procesa las operaciones de escritura de forma secuencial y se detiene ante el primer error que detecta. Esto resulta útil cuando el orden de las operaciones es importante, por ejemplo, cuando las operaciones posteriores dependen de las anteriores. Sin embargo, las operaciones ordenadas suelen ser más lentas que las desordenadas. En el caso de las operaciones ordenadas, debe abordar el caso de que el lote se detenga ante el primer error, lo que podría dejar algunas operaciones sin procesar.
- Operaciones desordenadas: permite a Amazon DocumentDB procesar las inserciones como una sola ejecución en la base de datos. Si se produce un error en un documento, la operación continúa con el resto de los documentos. Esto resulta especialmente útil cuando se insertan grandes cantidades de datos y se pueden tolerar algunos errores, como durante la migración de datos o las importaciones masivas, donde algunos documentos pueden fallar debido a la duplicación de claves. En el caso de operaciones desordenadas, debe abordar las situaciones de éxito parcial, en los que algunas operaciones se realizan correctamente y otras no.

Al trabajar con el método `bulkWrite()`, se requieren algunas clases esenciales. En primer lugar, la clase `WriteModel` sirve como clase base para todas las operaciones de escritura y tiene

implementaciones específicas como `InsertOneModel`, `UpdateOneModel`, `UpdateManyModel`, `DeleteOneModel` y `DeleteManyModel`, que manejan diferentes tipos de operaciones.

La `BulkWriteOptions` clase es necesaria para configurar el comportamiento de las operaciones masivas, como configurar la ordered/unordered ejecución o eludir la validación de documentos. La clase `BulkWriteResult` proporciona información detallada sobre los resultados de la ejecución, incluidos los recuentos de documentos insertados, actualizados y eliminados.

Para la gestión de errores, la clase `MongoBulkWriteException` es fundamental, ya que contiene información sobre los posibles errores que se produzcan durante la operación masiva, mientras que la clase `BulkWriteError` proporciona detalles específicos sobre los errores de las operaciones individuales. El siguiente código muestra un ejemplo de inserción de una lista de documentos, así como de actualización y eliminación de un solo documento, todo ello mediante la ejecución de una llamada a un único método `bulkWrite()`. El código también muestra cómo trabajar con `BulkWriteOptions` y `BulkWriteResult`, así como la correcta gestión de errores de la operación de `bulkWrite()`.

```
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());
}
```

Escrituras que se pueden volver a intentar

A diferencia de MongoDB, Amazon DocumentDB no admite el reintento de las escrituras. En consecuencia, debe implementar una lógica de reintento personalizada en sus aplicaciones, especialmente para solucionar problemas de red o la falta temporal de disponibilidad del servicio. Por lo general, una estrategia de reintentos bien implementada implica aumentar la demora entre los reintentos y limitar el número total de reintentos. Consulte Gestión de errores con lógica de reintento a continuación para conocer un ejemplo de código sobre cómo crear una lógica de reintentos con gestión de errores.

Lectura y recuperación de datos de una colección de DocumentDB

La consulta de documentos en Amazon DocumentDB gira en torno a varios componentes clave que permiten recuperar y manipular datos con precisión. El método `find()` es la API de consulta fundamental en los controladores Java de MongoDB. Permite la recuperación de datos complejos con numerosas opciones para filtrar, ordenar y proyectar resultados. Además del método `find()`, `Filters` y `FindIterable` son otros dos componentes fundamentales que proporcionan los componentes básicos para las operaciones de consulta en los controladores Java de MongoDB.

La clase `Filters` es una clase de utilidad del controlador Java de MongoDB que proporciona una API fluida para generar un constructo de filtros de consultas. Esta clase ofrece métodos de fábrica estáticos que crean instancias de objetos `Bson` que representan diversas condiciones de consulta.

Los métodos más comúnmente utilizados son `eq()` para las comparaciones de igualdad, `gt()`, `lt()`, `gte()` y `lte()` para las comparaciones numéricas, `and()` y `or()` para las combinaciones de condiciones múltiples, `in()` y `nin()` para las pruebas de suscripción a matrices y `regex()` para la coincidencia de patrones. La clase está diseñada para ser segura en relación con los tipos y proporciona una mejor comprobación en tiempo de compilación en comparación con las consultas basadas en documentos sin procesar, lo que la convierte en el enfoque preferido para crear consultas de DocumentDB en aplicaciones Java. La gestión de errores es sólida, con claras excepciones en el caso de constructos de filtros no válidas.

`FindIterable` es una interfaz especializada diseñada para gestionar el resultado del método `find()`. Proporciona un amplio conjunto de métodos para refinar y controlar la ejecución de consultas, y ofrece una API fluida para el encadenamiento de métodos. La interfaz incluye métodos esenciales de modificación de consultas, como `limit()` para restringir el número de documentos devueltos, `skip()` para paginar, `sort()` para ordenar los resultados, `projection()` para seleccionar campos específicos y `hint()` para seleccionar índices. Las operaciones de agrupar en lotes, omitir y limitar en `FindIterable` son herramientas esenciales de paginación y administración de datos que ayudan a controlar la forma en que se recuperan y procesan los documentos de la base de datos.

Agrupar en lotes (`batchSize`) controla el número de documentos que DocumentDB devuelve al cliente en una única red de ida y vuelta. Cuando establece un tamaño de lote, DocumentDB no devuelve todos los documentos coincidentes a la vez, sino que los devuelve en grupos del tamaño de lote especificado.

Omitir le permite desplazar el punto de partida de los resultados y, básicamente, le indica a DocumentDB que omita un número específico de documentos antes de empezar a devolver las coincidencias. Por ejemplo, `skip(20)` omitirá los primeros 20 documentos coincidentes. Esto se suele utilizar en situaciones de paginación en los que se desean recuperar páginas de resultados posteriores.

Limitar restringe el número total de documentos que se pueden devolver a partir de una consulta. Si especifica `limit(n)`, DocumentDB dejará de devolver documentos después de haber devuelto «n» documentos, incluso si hay más coincidencias en la base de datos.

`FindIterable` admite patrones de iterador y cursor cuando recupera documentos de Amazon DocumentDB. La ventaja de usar `FindIterable` como iterador es que permite la carga diferida de los documentos y solo los recupera cuando la aplicación los solicita. Otra ventaja de usar el iterador es que no es responsable de mantener la conexión con el clúster y, por lo tanto, no es necesario cerrar la conexión de forma explícita.

`FindIterable` también proporciona soporte para `MongoCursor`, que permite utilizar patrones de cursor cuando trabaja con consultas de Amazon DocumentDB. El `MongoCursor` es una implementación específica del controlador Java de MongoDB que proporciona control sobre las operaciones de la base de datos y la administración de recursos. Implementa la interfaz de `AutoCloseable`, lo que permite una gestión explícita de los recursos mediante bloques de prueba con los recursos. Esto es crucial para cerrar correctamente las conexiones de las bases de datos y liberar recursos del servidor. De forma predeterminada, el tiempo de espera del cursor es de 10 minutos y DocumentDB no le da la opción de cambiar este comportamiento de tiempo de espera. Cuando trabaje con datos agrupados, asegúrese de recuperar el siguiente lote de datos antes de que se agote el tiempo de espera del cursor. Una consideración clave a la hora de usar el `MongoCursor` es que requiere un cierre explícito para evitar la pérdida de recursos.

En esta sección, se presentan varios ejemplos de `find()`, `Filters` y `FindIterable`.

En el siguiente ejemplo de código se muestra cómo usar `find()` para recuperar un solo documento con su campo «restaurantId»:

```
Document filter = new Document("restaurantId", "REST-21G145");
Document result = collection.find(filter).first();
```

Si bien el uso de `Filters` permite comprobar mejor los errores en el momento de la compilación, el controlador java también permite especificar un filtro de `Bson` directamente en el método `find()`. El siguiente código de ejemplo pasa el documento `Bson` a `find()`:

```
result = collection.find(new Document("$and", Arrays.asList(
    new Document("rating.totalReviews", new Document("$gt", 1000)),
    new Document("priceRange", "$$"))))
```

El siguiente código de ejemplo muestra varios ejemplos del uso de la clase `Filters` 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));
```

En el siguiente ejemplo se muestra cómo encadenar las operaciones de `sort()`, `skip()`, `limit()` y `batchSize()` en un objeto de `FindIterable`. El orden en que se proporcionen estas operaciones influirá en el rendimiento de la consulta. Como práctica recomendada, el orden de estas operaciones debe ser `sort()`, `projection()`, `skip()`, `limit()` y `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);
```

El siguiente ejemplo de código muestra cómo crear un iterador en `FindIterable`. Usa el constructo `forEach` de Java para recorrer el conjunto de resultados.

```
collection.find(Filters.eq("cuisine", "American")).forEach(doc ->
 System.out.println(doc.toJson()));
```

En el último ejemplo de código de `find()`, se muestra cómo usar el `cursor()` para la recuperación de documentos. Crea el cursor en el bloque de prueba, lo que garantiza que el cursor se cierre cuando el código salga del bloque de prueba.

```
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
```

Actualización de documentos existentes en una colección de DocumentDB

Amazon DocumentDB proporciona mecanismos flexibles y potentes para modificar los documentos existentes e insertar otros nuevos cuando no existen. El controlador Java de MongoDB ofrece varios métodos de actualización: `updateOne()` para actualizaciones de un solo documento, `updateMany()` para actualizaciones de varios documentos y `replaceOne()` para la sustitución completa de documentos. Además de estos tres métodos, `Updates`, `UpdateOptions` y `UpdateResult` son otros componentes fundamentales que proporcionan los componentes básicos para las operaciones de actualización en los controladores Java de MongoDB.

La clase `Updates` del controlador Java de MongoDB es una clase de utilidad que proporciona métodos de fábrica estáticos para crear operadores de actualización. Sirve como el compilador principal para crear operaciones de actualización de forma legible y segura. Métodos básicos como `set()`, `unset()` y `inc()` permiten la modificación directa de los documentos. El poder de esta clase se hace evidente cuando se combinan varias operaciones mediante el método `Updates.combine()`, que permite ejecutar varias operaciones de actualización de forma atómica, lo que garantiza la coherencia de datos.

`UpdateOptions` es una potente clase de configuración del controlador Java de MongoDB que proporciona capacidades de personalización esenciales para las operaciones de actualización de documentos. Dos aspectos importantes de esta clase son la compatibilidad con actualizaciones y filtros de matriz para las operaciones de actualización. La característica de actualización, habilitada mediante `upsert(true)`, permite crear nuevos documentos cuando no se encuentra ningún documento coincidente durante una operación de actualización. Mediante `arrayFilters()`, la operación de actualización puede actualizar con precisión los elementos de la matriz que cumplan criterios específicos.

`UpdateResult` en el controlador Java de MongoDB proporciona el mecanismo de retroalimentación que detalla el resultado de una operación de actualización. Esta clase agrupa tres métricas clave: el número de documentos que cumplen los criterios de actualización (`matchedCount`), el número de documentos realmente modificados (`modifiedCount`) y la información sobre cualquier documento alterado (`upsertedId`). Comprender estas métricas es esencial para gestionar correctamente los errores, verificar las operaciones de actualización y mantener la coherencia de datos en las aplicaciones.

Actualización y sustitución de un solo documento

En DocumentDB, la actualización de un solo documento se puede realizar mediante el método updateOne(). Este método utiliza un parámetro de filtro, normalmente proporcionado por la clase `Filters`, para identificar el documento que se va a actualizar, un parámetro de `Updat` que determina qué campos se van a actualizar y un parámetro de `UpdateOptions` opcional para establecer diferentes opciones de actualización. El uso del método `updateOne()` solo actualizará el primer documento que coincida con los criterios de selección. El siguiente código de ejemplo actualiza un solo campo de un documento:

```
collection.updateOne(Filters.eq("restaurantId", "REST-Y2E9H5"),
    Updates.set("name", "Amazing Japanese sushi"));
```

Para actualizar varios campos de un documento, use `updateOne()` con `Update.combine()` como se muestra en el siguiente ejemplo. En este ejemplo también se muestra cómo agregar un elemento a una matriz del 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);
```

En el siguiente ejemplo de código se muestra cómo actualizar un documento de la base de datos. Si el documento especificado no existe, la operación lo insertará automáticamente como documento nuevo. Este código también muestra cómo utilizar las métricas disponibles a través del objeto de `UpdateResult`.

```
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)");
}
```

El siguiente ejemplo de código muestra cómo reemplazar completamente un documento existente por uno nuevo mediante el método `replaceOne()`, en lugar de actualizar campos individuales. El método `replaceOne()` sobrescribe todo el documento y conserva únicamente el campo `_id` del original. Si varios documentos coinciden con los criterios del filtro, solo se reemplaza el primer documento encontrado.

```
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());
```

Actualización de documentos múltiples

Hay dos formas de actualizar varios documentos de una colección de forma simultánea. Puede utilizar el método de `updateMany()` o usar el `UpdateManyModel` con el método `bulkWrite()`. El método `updateMany()` utiliza un parámetro de filtro para seleccionar los documentos que se van a actualizar, el parámetro `Update` para identificar los campos que se van a actualizar y un parámetro `UpdateOptions` opcional para especificar las opciones de actualización.

El siguiente ejemplo de código demuestra el uso del método `updateMany()`:

```
Bson filter = Filters.and(
    Filters.in("features", Arrays.asList("Private Dining")),
    Filters.eq("cuisine", "Thai"));
UpdateResult result1 = collection.updateMany(filter, Updates.set("priceRange", "$$$"));
```

El siguiente ejemplo de código demuestra el método de `bulkWrite()` con la misma actualización:

```
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());
```

Eliminación de documentos de una colección de DocumentDB

El controlador Java de MongoDB ofrece `deleteOne()` para eliminar un solo documento y `deleteMany()` para eliminar varios documentos que coincidan con criterios específicos. Al igual que la actualización, la operación de eliminación también se puede utilizar con el método `bulkWrite()`. Ambos `deleteOne()` y `deleteMany()` devuelven un objeto `DeleteResult` que proporciona información sobre el resultado de la operación, incluido el recuento de documentos eliminados. A continuación, se muestra un ejemplo de cómo usar `deleteMany()` para eliminar varios documentos:

```
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());
```

Gestión de errores con lógica de reintento

Una estrategia sólida de gestión de errores con Amazon DocumentDB debería incluir la categorización de los errores en reintentables (como tiempos de espera de la red o problemas de conexión) y no reintentables (como errores de autenticación o consultas no válidas). En el caso de los errores de operación debidos a errores que deben reintentarse, se debe implementar un intervalo de tiempo entre cada reintento, así como un máximo de reintentos. Las operaciones CRUD deben realizarse en un bloque try-catch que capture `MongoException` y sus subclases. Además, debe incluir la supervisión y el registro de los errores para garantizar la visibilidad operativa. El siguiente es un ejemplo de código que muestra cómo implementar la gestión de errores de reintentos:

```
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);
    }
}
```

Administración de índices en Amazon DocumentDB con Java

Los índices permiten una recuperación eficaz de los datos de una colección de Amazon DocumentDB. Sin índices, DocumentDB debe digitalizar todos los documentos de la colección para obtener resultados que satisfagan una consulta determinada. En este tema se proporciona información sobre cómo crear, eliminar y enumerar índices mediante los controladores Java de MongoDB. También se explica cómo determinar si se está utilizando un índice específico en la consulta y cómo hacer sugerencias a Amazon DocumentDB para que utilice un índice específico.

Temas

- Creación de índices con Java
- Eliminar índices
- Determinación de la selección del índice y proporción de una sugerencia sobre el índice

Amazon DocumentDB admite muchos tipos de índices. Para obtener una descripción general completa de todos los índices compatibles, consulte Cómo indexar en Amazon DocumentDB en el blog de bases de datos. AWS

Creación de índices con Java

Existen dos mecanismos para crear índices en Amazon DocumentDB con controladores Java de MongoDB: mediante `runCommand()`, y mediante el método `createIndex()` para un índice único o el método `createIndexes()` para varios índices. Uno de los motivos para utilizar los métodos `createIndex()` y `createIndexes()` es que permiten gestionar mejor los errores, ya que detecta errores específicos relacionados con la creación de índices. Otra razón para usar estos métodos en lugar de `runCommand()` es que el controlador Java de MongDB proporciona un amplio conjunto de clases admitidas para la creación y manipulación de índices. Tenga en cuenta que

estas clases admitidas pueden utilizarse solamente cuando use los métodos `createIndex()` o `createIndexes()`. Hay tres clases admitidas:

- **`Indexes`**: esta clase sirve como clase de utilidad y ofrece métodos de fábrica estáticos para crear varios tipos de índices. Simplifica el proceso de creación de definiciones de índices complejas y se suele usar junto con otras clases relacionadas con los índices.
- **`IndexModel`**: se trata de una clase fundamental que condensa tanto la definición de las claves de índice como sus opciones. Representa una especificación de índice completa, que combina qué indexar (las claves) con cómo indexar (las opciones). Esta clase es particularmente útil cuando se crean varios índices simultáneamente, ya que permite definir un conjunto de especificaciones de índices que se pueden pasar al método `createIndexes()`.
- **`IndexOptions`**: se trata de una clase de configuración completa que proporciona un amplio conjunto de métodos para personalizar el comportamiento de los índices. Incluye ajustes para índices únicos, índices dispersos, tiempo de caducidad (TTL) y expresiones de filtrado parcial. Mediante el encadenamiento de métodos, puede configurar varias opciones, como la creación de índices en segundo plano y las restricciones únicas.

Creación de un índice único

En este ejemplo se muestra cómo crear un índice único con el método `createIndex()` en segundo plano. Para comprender la creación de índices en segundo plano y en segundo plano, consulte. Tipos de creación de índices El siguiente ejemplo de código sirve `IndexOptions` para crear un índice único con el nombre «unique_restaurantId_idx» en segundo plano. A continuación, este objeto `IndexOptions` se pasa al método `createIndex()`.

```
collection.createIndex(
    Indexes.ascending("restaurantId"),
    new IndexOptions()
        .unique(true)
        .name("unique_restaurantId_idx")
        .background(true));
```

Creación de índices múltiples

En este ejemplo, se crean índices múltiples mediante el método `createIndexes()`. Primero crea la opción para cada índice mediante el objeto de `IndexModel` y, a continuación, pasa una lista de objetos de `IndexModel` al método `createIndexes()`. En el siguiente ejemplo de código se muestra cómo crear un índice compuesto mediante la clase de utilidad `Indexes`. Esta clase

también se usa para especificar si desea crear un índice con un orden de clasificación ascendente o descendente. Tras crear índices múltiples, llama al método `listIndexes()` para verificar la creación del índice.

```
// 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()));
```

Creación de índices dispersos y parciales

En este ejemplo se muestra la creación de un índice disperso y uno parcial mediante la creación de un `IndexModel` para cada tipo de índice.

```
// 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))));
```

Creación de un índice de texto

En este ejemplo se muestra cómo crear un índice de texto. Solo se permite un índice de texto en una colección, pero ese índice de texto puede ser un índice compuesto que abarque campos múltiples. Cuando usa varios campos en el índice de texto, también puede asignar pesos a cada uno de los campos del índice. Amazon DocumentDB no admite los índices de texto en campos de matriz y, aunque puede utilizar hasta 30 campos en el índice de texto compuesto, solo se puede asignar un peso a tres campos.

```
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());
```

Creación de un índice mediante el **`runCommand()`**

Amazon DocumentDB admite la creación de índices en paralelo para reducir el tiempo que se tarda en crear índices. La indexación paralela utiliza varios procesos de trabajo simultáneos. La cantidad predeterminada de procesos de trabajo que se utilizan para la creación de índices son dos. Para obtener más información, consulte Descubra el potencial de la indexación paralela en Amazon DocumentDB en el blog de bases de datos. AWS Actualmente, los controladores Java de MongDB no permiten especificar la opción de proceso de trabajo cuando se utiliza `createIndex()` o `createIndexes()` y, por lo tanto, la única forma de especificar los procesos de trabajo es mediante

el `runCommand`. El siguiente ejemplo de código muestra cómo usar `runCommand` para crear un índice que aumente el número de procesos de trabajo a cuatro:

```
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);
```

Eliminar índices

El controlador Java de MongoDB proporciona varios métodos para eliminar índices, ya que se adapta a diferentes situaciones y a sus preferencias. Puede eliminar los índices por nombre, por especificación de clave o eliminar todos los índices a la vez. Los métodos `dropIndex()` y `dropIndexes()` pueden invocarse en un objeto de colección para eliminar un índice. Cuando elimina un índice por nombre, debe asegurarse de que se use el nombre de índice correcto, lo que puede no ser siempre intuitivo, especialmente para los índices compuestos o generados de forma automática. Si intenta eliminar un índice inexistente, se obtendrá una `MongoCommandException`. El índice `default _id` no se puede eliminar, ya que garantiza la exclusividad del documento dentro de la colección.

En el siguiente ejemplo de código se muestra cómo eliminar un índice con el nombre del campo donde se creó el índice o mediante la eliminación de todos los índices:

```
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();
```

Cuando borra índices con claves múltiples, asegúrese de que haya un índice compuesto que contenga todas las claves especificadas y de que el orden de las claves sea correcto. El código

de ejemplo de creación del índice anterior muestra una clave compuesta sobre «cocina» y características. Si intenta eliminar esa clave compuesta pero el orden no es el que se usó para crearla, se produce MongoCommnadException el siguiente error:

```
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);
}
```

Se muestra el siguiente error:

```
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
```

Determinación de la selección del índice y proporción de una sugerencia sobre el índice

Trabajar con la funcionalidad de explicación de Amazon DocumentDB es fundamental para comprender el rendimiento de las consultas y el uso de los índices. Cuando ejecuta una consulta, puede añadir el método `explain()` para obtener información detallada sobre el plan de consulta, incluidos los índices, si los hubiera, que se están utilizando. El resultado `explain()` proporciona información sobre las etapas de ejecución de la consulta, el número de documentos examinados y el tiempo necesario para cada etapa. Esta información es inestimable para identificar si un índice en particular se está utilizando de manera eficaz o si la consulta podría beneficiarse de una estructura de índice diferente.

El método `explain()` se puede encadenar con el método `find()`. El método `explain()` puede utilizar un enumerador opcional `ExplainVerbosity` que determine el nivel de verbosidad devuelto por `explain()`. En este momento, DocumentDB admite solamente los enumeradores

EXECUTION_STATS y QUERY_PLANNER. En el siguiente ejemplo de código se muestra cómo obtener el planificador de consultas para una consulta específica:

```
// 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());
```

Se devuelve el siguiente documento JSON para el nivel de verbosidad del planificador de consultas:

```
{
  "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
    }
  }
}
```

Dispone de varias opciones para incitar o forzar a Amazon DocumentDB a utilizar un índice específico. Los métodos `hint()` y `hintString()` le permiten anular el comportamiento de selección de índices predeterminado del optimizador de consultas y especificar explícitamente qué

índice debe usarse para una consulta. Si bien el optimizador de consultas de DocumentDB suele ser una buena opción para la selección de índices, hay situaciones en las que forzar la aprobación de un índice específico mediante `hint()` o `hintString()` puede resultar beneficioso, como cuando se trata de datos sesgados o se comprueba el rendimiento de un índice.

El siguiente ejemplo de código fuerza el uso de un índice compuesto «cuisine_features_idx» para la misma consulta que se ejecutó en el código anterior:

```
// 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 programación con Amazon DocumentDB y Java

Event-driven la programación en el contexto de Amazon DocumentDB representa un poderoso patrón arquitectónico en el que los cambios en las bases de datos actúan como los principales generadores de eventos que activan la lógica y los procesos empresariales posteriores. Cuando se insertan, actualizan o eliminan registros en una colección de DocumentDB, estos cambios actúan como eventos que inician automáticamente varios procesos posteriores, notificaciones o tareas de sincronización de datos. Este patrón es particularmente valioso en los sistemas distribuidos modernos, donde aplicaciones o servicios múltiples deben reaccionar a los cambios en los datos en tiempo real. El mecanismo principal de implementación de la programación basada en eventos en DocumentDB es mediante flujos de cambios.

> **Note**
>
> En esta guía se asume que habilitó los flujos de cambios en una colección con la que está trabajando. Consulte Uso de flujos de cambios con Amazon DocumentDB para obtener información sobre cómo habilitar los flujos de cambios en la colección.

Trabajo con flujos de cambios desde la aplicación Java

El método `watch()` del controlador Java de MongoDB es el mecanismo principal para supervisar los cambios de datos en tiempo real en Amazon DocumentDB. Los objetos `MongoClient`, `MongoDatabase` y `MongoCollection` pueden llamar al método `watch()`.

El método `watch()` devuelve una instancia de `ChangeStreamIterable` que admite varias opciones de configuración, entre las que se incluyen la búsqueda completa de actualizaciones en los documentos, el suministro de tokens de reanudación y una marca de tiempo para garantizar la fiabilidad, y las etapas de agregación de canalizaciones para filtrar los cambios.

`ChangeStreamIterable` implementa la interfaz básica de Java `Iterable` y puede usarse con `forEach()`. Para capturar eventos mediante `forEach()`, transfiera una función de devolución de llamada a `forEach()` que procese el evento cambiado. En el siguiente fragmento de código se muestra cómo abrir un flujo de cambios en una colección para iniciar la supervisión de eventos de cambio:

```
ChangeStreamIterable < Document > iterator = collection.watch();
iterator.forEach(event - > {
    System.out.println("Received a change: " + event);
});
```

Otra forma de recorrer todos los eventos de cambio consiste en abrir un cursor que mantenga una conexión con el clúster y reciba continuamente nuevos eventos de cambio a medida que se producen. Para obtener un cursor de flujo de cambios, utilice el método `cursor()` del objeto `ChangeStreamIterable`. En el siguiente ejemplo de código se muestra cómo supervisar eventos de cambio con el cursor:

```
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().cursor()) {
    System.out.println(cursor.tryNext());
}
```

Como práctica recomendada, puede crear el `MongoChangeStreamCursor` en una declaración de prueba con recursos o cerrar el cursor manualmente. Si se llama al método `cursor()` en `ChangeStreamIterable`, se devuelve un `MongoChangeStreamCursor` creado sobre un objeto de `ChangeStreamDocument`.

La clase `ChangeStreamDocument` es un componente crucial que representa los eventos de cambio individuales en el flujo. Contiene información detallada sobre cada modificación, incluido el tipo de operación (insertar, actualizar, eliminar, reemplazar), la clave del documento,

la información del espacio de nombres y el contenido completo del documento, cuando esté disponible. La clase proporciona métodos para acceder a varios aspectos del evento de cambio, como `getOperationType()` para determinar el tipo de cambio, `getFullDocument()` para acceder al estado completo del documento y `getDocumentKey()` para identificar el documento modificado.

El objeto `ChangeStreamDocument` proporciona dos datos importantes: un token de reanudación y la hora del evento de cambio.

Los tokens de reanudación y las operaciones basadas en el tiempo en los flujos de cambios de DocumentDB proporcionan mecanismos cruciales para mantener la continuidad y administrar el acceso a los datos históricos. Un token de reanudación es un identificador único que se genera para cada evento de cambio y sirve como marcador que permite que las aplicaciones reinicien el procesamiento del flujo de cambios desde un punto específico tras una desconexión o un fallo. Cuando se crea un cursor de flujo de cambios, se puede usar un token de reanudación previamente almacenado mediante la opción `resumeAfter()`, lo que permite que el flujo continúe desde donde dejó en lugar de empezar desde el principio o perder los eventos.

Time-based las operaciones en los flujos de cambio ofrecen diferentes enfoques para gestionar el punto de partida de la supervisión de los eventos de cambio. La opción `startAtOperationTime()` le permite empezar a observar los cambios que ocurrieron en una marca de tiempo específica o después de esta. Estas funciones basadas en el tiempo son particularmente valiosas en situaciones que requieren el procesamiento de datos históricos, la recuperación en un momento dado o la sincronización entre sistemas.

El siguiente ejemplo de código recupera el evento asociado al documento de inserción, captura su token de reanudación y, a continuación, proporciona ese token para empezar a supervisar los eventos posteriores al evento de inserción. El evento se asocia al evento de actualización y, a continuación, obtiene la hora del clúster en que se produjo la actualización y utiliza esa marca de tiempo como punto de partida para el procesamiento posterior.

```
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());
  }
```

De manera predeterminada, el evento de cambio de actualización no incluye el documento completo, sino tan solo los cambios realizados. Si necesita acceder al documento completo que se actualizó, puede llamar al método `fullDocument()` en el objeto `ChangeStreamIterable`. Tenga en cuenta que, cuando solicita que se devuelva un documento completo para un evento de actualización, se devolverá el documento que existía en el momento en que se realizó la llamada al flujo de cambios.

Este método toma un enumerador `FullDocument` como parámetro. Actualmente, Amazon DocumentDB admite solamente valores DEFAULT y `UPDATE_LOOKUP`. El siguiente fragmento de código muestra cómo solicitar el documento completo para los eventos de actualización cuando empieza a observar los cambios:

```
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().fullDocument(FullDocument.UPDATE_LOOKUP).cursor())
```

Cómo utilizar la validación de esquemas JSON

Con el operador de consulta de evaluación de `$jsonSchema`, puede validar los documentos que se están insertando en sus colecciones.

Temas

- Cómo crear y utilizar la validación de esquemas JSON
- Palabras clave compatibles

- circunvalación DocumentValidation
- Limitaciones

Cómo crear y utilizar la validación de esquemas JSON

Cómo crear una colección con validación de esquemas

Puede crear una colección con reglas de operación y validación de `createCollection`. Estas reglas de validación se aplican al insertar o actualizar documentos de Amazon DocumentDB. En el siguiente ejemplo de código se muestran las reglas de validación para un conjunto de empleados:

```
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"
} )
```

Cómo insertar un documento válido

En el siguiente ejemplo, se insertan documentos que cumplen con las reglas de validación de esquemas anteriores:

```
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})
```

Cómo insertar un documento no válido

En el siguiente ejemplo, se insertan documentos que no cumplen con las reglas de validación de esquemas anteriores. En este ejemplo, el valor employeeId no es una cadena:

```
db.employees.insert({
    "name" : { "firstName" : "Carol" , "lastName" : "Smith"},
    "employeeId": 720 ,
    "salary": 1000.0
})
```

En este ejemplo, se muestra una sintaxis incorrecta en el documento.

Modificación de una colección

El comando `collMod` se utiliza para añadir o modificar las reglas de validación de la colección existente. En el siguiente ejemplo se añade un campo de salario a la lista de campos obligatorios:

```
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
      }
   }
} )
```

Cómo abordar los documentos añadidos antes de que se cambiaran las reglas de validación

Para abordar los documentos que se añadieron a su colección antes de que se cambiaran las reglas de validación, utilice los siguientes modificadores de `validationLevel`:

- estricto: aplica reglas de validación a todas las inserciones y actualizaciones.
- moderado: aplica reglas de validación a los documentos válidos existentes. Durante las actualizaciones, no se comprueban los documentos no válidos existentes.

En el siguiente ejemplo, tras actualizar las reglas de validación de la colección denominada “empleados”, el campo de salario es obligatorio. No se podrá actualizar el siguiente documento:

```
db.runCommand({
    update: "employees",
    updates: [{
        q: { "employeeId": "c721a" },
```

```
        u: { age: 25 , salary : 1000},
        upsert: true }]
})
```

Amazon DocumentDB devuelve el siguiente resultado:

```
{
"n" : 0,
    "nModified" : 0,
    "writeErrors" : [
        {
"index" : 0,
            "code" : 121,
            "errmsg" : "Document failed validation"
        }
    ],
    "ok" : 1,
    "operationTime" : Timestamp(1234567890, 1)
}
```

Si se actualiza el nivel de validación a `moderate`, se permitirá actualizar el documento anterior correctamente:

```
db.runCommand({
    "collMod" : "employees",
    validationLevel : "moderate"
})

db.runCommand({
    update: "employees",
    updates: [{
        q: { "employeeId": "c721a" },
        u: { age: 25 , salary : 1000},
        upsert: true }]
})
```

Amazon DocumentDB devuelve el siguiente resultado:

```
{
"n" : 1,
    "nModified" : 1,
    "ok" : 1,
```

```
    "operationTime" : Timestamp(1234567890, 1)
}
```

Recuperación de documentos con el $jsonSchema

El operador de `$jsonSchema` se puede utilizar como filtro para consultar documentos que coincidan con el esquema JSON. Se trata de un operador de nivel superior que puede estar presente en los documentos de filtro como campo de nivel superior o utilizarse con operadores de consulta como `$and`, `$or` y `$nor`. En los siguientes ejemplos se muestra el uso de $jsonSchema como filtro individual y con otros operadores de filtro:

Documento insertado en una colección de “empleado”:

```
{ "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 }
```

Colección filtrada únicamente con el operador de `$jsonSchema`:

```
db.employees.find({
       $jsonSchema: { required: ["age"] } })
```

Amazon DocumentDB devuelve el siguiente resultado:

```
{ "_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 }
```

Colección filtrada con el operador de `$jsonSchema` y otro operador:

```
db.employees.find({
       $or: [{ $jsonSchema: { required: ["age", "name"]}},
            { salary: { $lte:1000}}]});
```

Amazon DocumentDB devuelve el siguiente resultado:

```
{ "_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 }
```

Colección filtrada con el operador de `$jsonSchema` y con `$match` en el filtro de agregación:

```
db.employees.aggregate(
    [{ $match: {
        $jsonSchema: {
            required: ["name", "employeeId"],
            properties: {"salary" :{"bsonType": "double"}}
        }
       }
    }]
)
```

Amazon DocumentDB devuelve el siguiente resultado:

```
{
"_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
}
```

Cómo visualizar las reglas de validación existentes

Para ver las reglas de validación existentes en una colección, utilice:

```
db.runCommand({
    listCollections: 1,
    filter: { name: 'employees' }
})
```

Amazon DocumentDB devuelve el siguiente resultado:

```
{
    "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 también conserva las reglas de validación en la etapa de agregación `$out`.

Palabras clave compatibles

Los comandos `create` y `collMod` admiten los siguientes campos:

- **`Validator`**: admite el operador `$jsonSchema`.
- **`ValidationLevel`**: admite los valores `off`, `strict` y `moderate`.
- **`ValidationAction`**: admite el valor `error`.

El operador $jsonSchema admite las siguientes palabras clave:

- `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`

circunvalación DocumentValidation

Amazon DocumentDB admite `bypassDocumentValidation` para los siguientes comandos y métodos:

- `insert`
- `update`
- `findAndModify`
- Etapa `$out` en el comando `aggregate` y en el método `db.collection.aggregate()`

Amazon DocumentDB no admite los siguientes comandos para `bypassDocumentValidation`:

- `$merge` en el comando `aggregate` y en el método `db.collection.aggregate()`
- Comando `mapReduce` y método `db.collection.mapReduce()`
- `applyOps` command

Limitaciones

Las siguientes limitaciones se aplican a la validación de `$jsonSchema`:

- Amazon DocumentDB devuelve el error “No se pudo validar el documento” cuando una operación no cumple con la regla de validación.
- Los clústeres elásticos de Amazon DocumentDB no admiten `$jsonSchema`.

Consultas en Amazon DocumentDB

En esta sección se explican todos los aspectos de las consultas con Amazon DocumentDB.

Temas

- Consulta de documentos
- Plan de consulta
- Descripción de los resultados
- Planificador de consultas v2
- Planificador de consultas v3
- Consulta de datos geoespaciales con Amazon DocumentDB
- Índice parcial
- Búsquedas de texto con Amazon DocumentDB

Consulta de documentos

A veces, es posible que tenga que examinar el inventario de su tienda online para que los clientes puedan ver y comprar lo que usted vende. Consultar una colección es relativamente fácil, tanto si desea consultar todos los documentos de la colección como solo aquellos que cumplan un determinado criterio.

Para consultar documentos, utilice la operación `find()`. El comando `find()` tiene un único parámetro de documento que define los criterios que se utilizan al elegir los documentos que se devuelven. El resultado de `find()` es un documento formateado como una sola línea de texto sin saltos de línea. Para formatear el documento resultante para facilitar su lectura, utilice `find().pretty()`. En todos los ejemplos que se muestran en este tema, se utiliza `.pretty()` para formatear la salida.

Los siguientes ejemplos de código utilizan los cuatro documentos que insertó en la `example` colección en los dos ejercicios anteriores `insertOne()` y `insertMany()` que se encuentran enAdición de documentos.

Temas

- Recuperar todos los documentos de una colección
- Recuperación de documentos que coincidan con un valor de campo

- Recuperación de documentos que coincidan con un documento incrustado
- Recuperación de documentos que coincidan con un valor de campo de un documento incrustado
- Recuperación de documentos que coincidan con una matriz
- Recuperación de documentos que coincidan con un valor de una matriz
- Recuperación de documentos mediante operadores

Recuperar todos los documentos de una colección

Para recuperar todos los documentos de la colección, utilice la operación `find()` con un documento de consulta vacío.

La siguiente consulta devuelve todos los documentos de la colección `example`.

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

Recuperación de documentos que coincidan con un valor de campo

Para recuperar todos los documentos que coincidan con un campo y valor, utilice la operación `find()` con un documento de consulta que identifique los campos y valores que desee.

Si se utilizan los documentos anteriores, esta consulta devuelve todos los documentos cuyo campo "Item" contiene "Pen".

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

Recuperación de documentos que coincidan con un documento incrustado

Para buscar todos los documentos que coinciden con un documento incrustado, utilice la operación `find()` con un documento de consulta que especifique el nombre del documento incrustado y todos los campos y los valores de ese documento incrustado.

Cuando se buscan coincidencias con un documento incrustado, el documento incrustado del documento debe tener el mismo nombre que en la consulta. Además, los campos y los valores del documento incrustado deben coincidir con la consulta.

La siguiente consulta devuelve únicamente el documento "Poster Paint". Esto se debe a que "Pen" tiene diferentes valores para "`MinOnHand`" y "`OnHand`", y "Spray Paint" tiene un campo más (`OrderQnty`) que el documento de consulta.

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

Recuperación de documentos que coincidan con un valor de campo de un documento incrustado

Para buscar todos los documentos que coinciden con un documento incrustado, utilice la operación `find()` con un documento de consulta que especifique el nombre del documento incrustado y todos los campos y los valores de ese documento incrustado.

Dados los documentos anteriores, la siguiente consulta utiliza la "notación de puntos" para especificar el documento incrustado y los campos de interés. Se devolverá cualquier documento que coincida con ellos, independientemente de los otros campos que puedan existir en el documento incrustado. La consulta devuelve "Poster Paint" y "Spray Paint", ya que ambos coinciden con los campos y los valores especificados.

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

Recuperación de documentos que coincidan con una matriz

Para buscar todos los documentos que coincidan con una matriz, utilice la operación `find()` con el nombre de la matriz que le interese y todos los valores de esa matriz. La consulta devuelve todos los documentos que tengan una matriz con ese nombre y cuyos valores de la matriz sean idénticos y estén en el mismo orden que en la consulta.

La siguiente consulta devuelve únicamente "Pen", ya que "Poster Paint" tiene un color adicional (White) y "Spray Paint" tiene los colores en otro orden.

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

Recuperación de documentos que coincidan con un valor de una matriz

Para buscar todos los documentos que tengan un valor determinado en una matriz, utilice la operación `find()` con el nombre de la matriz y el valor que le interese.

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

La operación anterior devuelve los tres documentos, ya que cada uno de ellos tiene una matriz denominada `Colors` y el valor "`Red`" en algún lugar de la matriz. Si especifica el valor "`White`", la consulta solo devolvería "Poster Paint".

Recuperación de documentos mediante operadores

La siguiente consulta devuelve todos los documentos en los que el valor de "`Inventory.OnHand`" es menor que 50.

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

Para obtener una lista de los operadores de consulta admitidos, consulte Operadores de consulta y proyección.

Plan de consulta

¿Cómo puedo ver las **`ExecutionStats de un`** plan de consulta?

Al determinar por qué una consulta se está ejecutando más lentamente de lo esperado, puede ser útil comprender qué `executionStats` son para el plan de consulta. `executionStats` proporciona el número de documentos devueltos desde una etapa determinada (`nReturned`), la cantidad de tiempo de ejecución empleado en cada etapa (`executionTimeMillisEstimate`) y la cantidad de tiempo que tarda en generar un plan de consulta (`planningTimeMillis`). Puede determinar las etapas de la consulta que requieren más tiempo para centrar sus esfuerzos de optimización en función del resultado`executionStats`, tal y como se muestra en los siguientes ejemplos de consultas. El parámetro `executionStats` actualmente no admite comandos `update` ni `delete`.

> **Note**
>
> Amazon DocumentDB emula la API MongoDB 3.6 en un motor de base de datos personalizada específicamente que utiliza un sistema de almacenamiento distribuido, tolerante a fallos y de recuperación automática. Como resultado, los planes de consulta y la salida de `explain()` pueden diferir entre Amazon DocumentDB y MongoDB. Los clientes que deseen controlar su plan de consulta pueden utilizar el operador `$hint` para aplicar la selección de un índice preferido.

Ejecute la consulta que desee mejorarse en el comando `explain()` del modo siguiente.

```
db.runCommand({explain: {query document}}).
explain("executionStats").executionStats;
```

A continuación se muestra un ejemplo de operación.

```
db.fish.find({}).limit(2).explain("executionStats");
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "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
}
```

Si está interesado en ver sólo el `executionStats` de la consulta anterior, puede utilizar el siguiente comando. En colecciones pequeñas, el procesador de consultas de Amazon DocumentDB puede optar por no utilizar un índice si las ventajas para el rendimiento son insignificantes.

```
db.fish.find({}).limit(2).explain("executionStats").executionStats;
```

Memoria caché de planes de consultas

Para optimizar el rendimiento y reducir la duración de la planificación, Amazon DocumentDB almacena internamente en caché los planes de consultas. Esto permite que las consultas con la misma forma se ejecuten directamente mediante un plan en caché.

Sin embargo, este almacenamiento en caché a veces puede provocar un retraso aleatorio para la misma consulta; por ejemplo, una consulta que normalmente tarda un segundo en ejecutarse puede tardar en ocasiones diez segundos. Esto se debe a que, con el tiempo, la instancia del lector almacenó en caché varias formas de la consulta, lo que consumió memoria. Si experimenta esta lentitud aleatoria, no es necesario que realice ninguna acción para liberar la memoria: el sistema gestionará el uso de la memoria por usted y, una vez que la memoria alcance un determinado umbral, se liberará automáticamente.

Descripción de los resultados

Si desea devolver información sobre los planes de consultas, Amazon DocumentDB admite el modo verbosidad `queryPlanner`. Los resultados `explain` devuelven el plan de consultas seleccionado elegido por el optimizador en un formato similar al siguiente:

```
{
   "queryPlanner" : {
      "plannerVersion" : <int>,
      "namespace" : <string>,
```

```
      "winningPlan" : {
         "stage" : <STAGE1>,
         ...
         "inputStage" : {
            "stage" : <STAGE2>,
            ...
            "inputStage" : {
               ...
            }
         }
      }
   }
}
```

En las siguientes secciones se definirán los resultados `explain` comunes.

Temas

- Etapa de escaneo y filtrado
- Intersección de índices
- Unión de índices
- Índice múltiple intersection/union
- Índice compuesto
- Etapa de clasificación
- Etapa de grupos

Etapa de escaneo y filtrado

El optimizador puede elegir uno de los siguientes escaneos:

COLLSCAN

Esta etapa es un escaneo secuencial de la colección.

```
{
    "stage" : "COLLSCAN"
}
```

IXSCAN

Esta etapa escanea las claves de índice. El optimizador puede recuperar el documento en esta etapa y esto puede provocar que se añada una etapa FETCH más adelante.

```
db.foo.find({"a": 1})
{
    "stage" : "IXSCAN",
    "direction" : "forward",
    "indexName" : <idx_name>
}
```

FETCH

Si el optimizador recuperó documentos en una etapa distinta de IXSCAN, el resultado incluirá una etapa FETCH. Por ejemplo, la consulta IXSCAN anterior puede resultar en una combinación de las etapas FETCH e IXSCAN:

```
db.foo.find({"a": 1})
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXSCAN",
        "indexName" : <idx_name>
    }
}
```

IXONLYSCAN escanea solo la clave de índice. Crear índices compuestos no evitará FETCH.

Intersección de índices

IXAND

Amazon DocumentDB puede incluir una etapa IXAND con una matriz InputStages de IXSCAN si puede utilizar la intersección de índices. Por ejemplo, podemos ver un resultado como el siguiente:

```
{
    "stage" : "FETCH",
    "inputStage" : {
```

```
        "stage" : "IXAND",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "indexName" : "a_1"
            },
            {
                "stage" : "IXSCAN",
                "indexName" : "b_1"
            }
        ]
    }
}
```

Unión de índices

MIXOR

De forma similar a la intersección de índices, Amazon DocumentDB puede incluir una etapa `IXOR` con una matriz `inputStages` para el operador `$or`.

```
db.foo.find({"$or": [{"a": {"$gt": 2}}, {"b": {"$lt": 2}}]})
```

Para la consulta anterior, el resultado de la explicación podría tener este aspecto:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "indexName" : "a_1"
            },
            {
                "stage" : "IXSCAN",
                "indexName" : "b_1"
            }
        ]
    }
}
```

Índice múltiple intersection/union

Amazon DocumentDB puede combinar varias etapas de unión o intersección de índices y, a continuación, obtener el resultado. Por ejemplo:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                ...
            },
            {
                "stage" : "IXAND",
                "inputStages" : [
                    {
                        "stage" : "IXSCAN",
                        ...
                    },
                    {
                        "stage" : "IXSCAN",
                        ...
                    }
                ]
            }
        ]
    }
}
```

El uso de las etapas de intersección o unión del índice no se ve afectado por el tipo de índice (escaso, compuesto, etc.).

Índice compuesto

El uso del índice compuesto de Amazon DocumentDB no está limitado a los subconjuntos iniciales de los campos indexados; puede utilizar el índice con la parte del sufijo, pero puede que no sea muy eficaz.

Por ejemplo, el índice compuesto de `{ a: 1, b: -1 }` puede admitir las tres consultas siguientes:

```
db.orders.find( { a: 1 } )

db.orders.find( { b: 1 } )

db.orders.find( { a: 1, b: 1 } )
```

Etapa de clasificación

Si hay un índice en las claves de clasificación solicitadas, Amazon DocumentDB puede usar el índice para obtener la clasificación. En ese caso, el resultado no incluirá una etapa `SORT`, sino más bien una etapa `IXSCAN`. Si el optimizador prefiere una ordenación simple, incluirá una etapa como esta:

```
{
    "stage" : "SORT",
    "sortPattern" : {
        "a" : 1,
        "b" : -1
    }
}
```

Etapa de grupos

Amazon DocumentDB admite dos estrategias de grupo diferentes:

- `SORT_AGGREGATE`: agregado de clasificación en disco.
- `HASH_AGGREGATE`: agregado de hash en memoria.

Planificador de consultas v2

El nuevo planificador de consultas para Amazon DocumentDB (versión 2.0 del planificador) incluye características avanzadas de optimización de consultas y un rendimiento mejorado. La versión 2.0 del planificador para Amazon DocumentDB 5.0 mejora el rendimiento hasta 10 veces con respecto a la versión anterior cuando se utilizan los operadores `find` y `update` con índices. Las mejoras en el rendimiento se deben principalmente al uso de planes de indexación más óptimos y a que se habilita el escaneo de índices para operadores como los operadores de negación (`$neq`, `$nin`) y anidados `$elementMatch`. Las consultas de la versión 2.0 del planificador se ejecutan más rápido gracias a mejores técnicas de estimación de costos, algoritmos optimizados y una estabilidad mejorada. La

versión 2.0 del planificador también es compatible con las API de filtrado de caché del plan, lo que mejora la estabilidad del planificador. Con esta característica, Amazon DocumentDB 5.0 ahora ofrece la posibilidad de seleccionar entre las distintas versiones del planificador de consultas.

Temas

- Requisitos previos
- Selección de la versión 2.0 del planificador como planificador de consultas predeterminado
- Prácticas recomendadas
- Limitaciones
- Mejoras en la búsqueda y actualización de operadores
- Planifique los filtros de caché
- Posibles diferencias de comportamiento entre el planificador, versión 1.0, 2.0 y MongoDB
- La versión 2.0 del planificador cierra la brecha de comportamiento con MongoDB

Requisitos previos

Para usar la versión 2.0 del planificador, se deben cumplir los siguientes requisitos previos:

- La versión 2.0 del planificador está disponible en todos los Regiones de AWS países donde está disponible la versión 5.0 del motor.
- Para optar por utilizar la versión 2.0 como planificador de consultas predeterminado, el clúster debe tener el parche del motor 3.0.15902 o posterior de la versión 5.0 de Amazon DocumentDB. Para obtener información sobre los pasos para actualizar el parche a la última versión del motor, consulte Actualización de un parche a la versión del motor de un clúster.
- Para configurar la versión 2.0 del planificador como el planificador de consultas predeterminado, necesita permisos de IAM para actualizar los grupos de parámetros del clúster.
- Para Amazon DocumentDB 8.0, la versión 3.0 del planificador de consultas predeterminado es el planificador de consultas predeterminado. Para obtener más información, consulte Planificador de consultas v3.

Selección de la versión 2.0 del planificador como planificador de consultas predeterminado

Siga los siguientes pasos para seleccionar 2.0 como el planificador de consultas predeterminado desde la consola o la CLI:

- Siga los pasos en Modificación de parámetros de clúster de Amazon DocumentDB para modificar el grupo de parámetros del clúster.
- Para el parámetro denominado «plannerVersion», cambie el valor a 2.0 para indicar la versión 2.0 del planificador.
- Seleccione Aplicar inmediatamente (si selecciona Aplicar al reiniciar, la selección no será válida hasta el siguiente reinicio del clúster).

Prácticas recomendadas

Para obtener los resultados esperados, utilice las siguientes prácticas recomendadas cuando aplique la versión 2.0 del planificador:

- En un clúster global, seleccione el mismo `plannerVersion` valor (1.0 o 2.0) en los grupos de parámetros del clúster para ambas regiones. Ten en cuenta que seleccionar diferentes versiones del planificador en las regiones principales y secundarias puede provocar que el comportamiento y el rendimiento de las consultas sean inconsistentes.
- La actualización a la versión 2.0 del planificador durante un período de mantenimiento programado o durante períodos de tráfico reducido será lo menos molesto, ya que puede haber un aumento en las tasas de error si se cambia la versión del planificador cuando las cargas de trabajo se están ejecutando activamente.
- La versión 2.0 del planificador funciona de manera óptima con el intérprete de comandos de MongoDB, versión 5.0.

Limitaciones

Las siguientes limitaciones se aplican a la versión 2.0 del planificador:

- La versión 2.0 del planificador no es compatible con los clústeres elásticos, que recurrirán a la versión 1.0 del planificador.
- La versión 2.0 del planificador no admite la agregación ni los comandos distintos, por lo que se recurrirá a la versión 1.0 del planificador.
- Algunas formas de consulta no pueden tener establecido un filtro de caché de planes. Para ver la lista actual, consulteFormas de consulta compatibles.

Mejoras en la **búsqueda** y **actualización de** operadores

La versión 2.0 del planificador optimiza las operaciones fundamentales, incluidos los comandos `find`, `update`, `delete` y `find-and-modify`. Las siguientes secciones con pestañas muestran las capacidades mejoradas de los índices, así como las mejoras en el rendimiento de las consultas con la versión 2.0 del planificador:

Enhanced index support

- La versión 2.0 del planificador añade compatibilidad con índices para los operadores de negación, incluidos `$nin`, `$ne`, `$not {eq}`, `$not {in}`, `$type` y `$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 versión 2.0 del planificador utiliza índices dispersos o parciales incluso cuando `$exists` no está presente en la expresión de consulta.

```
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 versión 2.0 del planificador utilizará índices parciales incluso cuando la condición de consulta no coincida exactamente con la expresión del filtro de índice parcial.

```
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 versión 2.0 del planificador utiliza un escaneo parcial del índice con consultas de $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 versión 2.0 del planificador incluye soporte con escaneo de índices para `$regex`, sin necesidad de incluir `$hint` en el código de la aplicación. `$regex` solo admite búsquedas de indexación por prefijo.

```
Sample Document: { "x": [1, 2, 3], "y": "apple" }
db.foo.createIndex({ "x": 1, "y": 1 })
db.foo.find({"y":{ $regex: "^a" }})
```

- La versión 2.0 del planificador mejora el rendimiento de las consultas que utilizan índices de claves múltiples, con condiciones de igualdad en los campos de claves múltiples.

```
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 versión 2.0 del planificador mejora el rendimiento de las consultas que utilizan filtros múltiples, especialmente en colecciones con documentos de más de 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 versión 2.0 del planificador mejora el rendimiento de las consultas cuando se utiliza el operador `$in` con un índice compuesto al eliminar la etapa de clasificación.

```
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})
```

También mejora el rendimiento de las consultas que utilizan índices de varias claves con `$in` elementos.

```
Sample Document: {"x": [1, 2, 3]}
db.foo.createIndex({"x": 1})
db.foo.find("x":{$in:[>100 elements]})
```

Query performance improvements

- La versión 2.0 del planificador mejora el rendimiento de las consultas que utilizan filtros múltiples, especialmente en colecciones con documentos de más de 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 versión 2.0 del planificador mejora el rendimiento de las consultas cuando se utiliza el operador `$in` con un índice compuesto al eliminar la etapa de clasificación.

```
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})
```

También mejora el rendimiento de las consultas que utilizan índices de claves múltiples con elementos `$in`.

```
Sample Document: {"x": [1, 2, 3]}
db.foo.createIndex({"x": 1})
db.foo.find("x":{$in:[>100 elements]})
```

Planifique los filtros de caché

La versión 2.0 de Planner añade compatibilidad con los filtros de caché del plan, también denominados filtros de índice, que permiten restringir el conjunto de índices que el planificador considera para una forma de consulta específica. Los filtros se configuran con un comando de base de datos y se aplican en el servidor, de modo que, si se produce una regresión en las consultas, puede mitigarla sin modificar el código de la aplicación.

En la versión 2.0 del planificador, los filtros de la caché del plan se aplican a `count` los comandos `find` y y, a partir de Amazon DocumentDB 5.0.2 y 8.0.2, también a los comandos `updatedelete`, y. `findAndModify` Los `aggregate` comandos `distinct` y requieren la versión 3.0 del planificador.

Para obtener más información sobre la referencia de comandos, las formas de consulta compatibles, los ejemplos prácticos y la forma en que los filtros interactúan con las sugerencias, consultePlanificar filtros de caché.

Posibles diferencias de comportamiento entre el planificador, versión 1.0, 2.0 y MongoDB

En algunos casos extremos, es posible que la versión 2.0 del planificador produzca resultados que varíen ligeramente de los resultados de MongoDB. En esta sección se describen algunos ejemplos de estas posibilidades.

$(update) and $(projection)

- En algunos casos, los operadores `$(update)` y `$(projection)` de MongoDB pueden comportarse de forma diferente a la versión 1.0 del planificador de Amazon DocumentDB. A continuación se muestran algunos ejemplos:

```
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 } }
```

- Planificador, versión 1.0: actualiza el campo 2022.
- MongoDB: actualiza el campo 2022.
- Planificador, versión 2.0: actualiza el campo 2021.

- ```
  db.col.insert({x:[1,2,3]})
  db.col.update({$and:[{x:1},{x:3}]},{$set:{"x.$":500}})
  ```
  - Planificador, versión 1.0: actualiza aleatoriamente el primer elemento coincidente.
  - MongoDB: actualiza aleatoriamente el primer elemento coincidente.
  - Planificador, versión 2.0: no realiza actualizaciones.
- ```
  db.col.insert({x:[1,2,3]})
  db.col.find()
  ```
  - Planificador, versión 1.0: selecciona aleatoriamente el elemento coincidente.
  - MongoDB: selecciona aleatoriamente el elemento coincidente.
  - Planificador, versión 2.0: no realiza ninguna selección.
- ```
  db.col.insert({x:100})
  db.col.update({x:100},{x:100})
  ```
  - Planificador, versión 1.0: cambios en el recuento nModified.
  - MongoDB: cambios en el recuento nModified.
  - Planificador, versión 2.0: el recuento nModified no cambia cuando se actualiza con el mismo valor.
- Cuando se usa el operador `$(update)` con `$setOnInsert`, la versión 1.0 del planificador y MongoDB arrojan un error, pero la versión 2.0 del planificador no.
- Si se cambia el nombre de un campo que no existe a `$field`, se produce un error en la versión 2.0 del planificador, mientras que no se producen actualizaciones en la versión 1.0 del planificador ni en MongoDB.
  ```

Index behavior

- La versión 2.0 del planificador arroja un error cuando se aplica `$hint` con un índice inadecuado, mientras que la versión 1.0 del planificador y MongoDB no lo hacen.

```
// Insert
db.col.insert({x:1})
db.col.insert({x:2})
db.col.insert({x:3})

// Create index on x with partialFilter Expression {x:{$gt:2}}
db.col.createIndex({x:1},{partialFilterExpression:{x:{$gt:2}}})

// Mongodb allows hint on the following queries
db.col.find({x:1}).hint("x_1")
// result is no documents returned because {x:1} is not indexed by the partial
 index
// Without $hint mongo should return {x:1}, thus the difference in result between
 COLSCAN and IXSCAN

DocumentDB will error out when $hint is applied on such cases.
db.col.find({x:1}).hint("x_1")
Error: error: {
    "ok" : 0,
    "operationTime" : Timestamp(1746473021, 1),
    "code" : 2,
    "errmsg" : "Cannot use Hint for this Query. Index is multi key index , partial
 index or sparse index and query is not optimized to use this index."
}

rs0:PRIMARY> db.runCommand({"planCacheSetFilter": "col", "query": { location:
 {$nearSphere: {$geometry: {type: "Point", coordinates: [1, 1]}}}}, "indexes":
 ["name_1"]})
{
    "ok" : 0,
    "operationTime" : Timestamp(1750815778, 1),
    "code" : 303,
    "errmsg" : "Unsupported query shape for index filter $nearSphere"
}
```

- `$near` no puede utilizar `$hint({“$natural”:1})` en la versión 2.0 del planificador.

```
// 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" }
```

- Si bien MongoDB admite escaneos completos de índices de expresiones regulares, la versión 2.0 del planificador admite el escaneo de índices de expresiones regulares solo en los campos con prefijo.

```
// 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)
```

- Existen algunas diferencias inherentes al uso de filtros de caché del plan con la versión 2.0 del planificador en comparación con MongoDB. Si bien la versión 2.0 del planificador no admite especificar la «proyección» y la «intercalación» con los filtros de caché del plan, MongoDB sí lo hace. Sin embargo, el filtro de índice de MongoDB solo está en memoria y se pierde tras el reinicio. La versión 2.0 del planificador conserva los filtros de índice mediante reinicios y parches.

Others

- El formato de los registros de auditoría de DML cuando se utiliza la versión 2.0 del planificador varía ligeramente de la versión 1.0.

```
command -  db.col.find({x:1})

************** Audit logs generated ******************

// v1 format for dml audit logs
{"atype":"authCheck","ts":1746473479983,"timestamp_utc":"2025-05-05
 19:31:19.983","remote_ip":"127.0.0.1:47022","users":
[{"user":"serviceadmin","db":"test"}],"param":
{"command":"find","ns":"test.col","args":{"batchSize":101,"filter":
{"x":1},"find":"col","limit":18446744073709551615,"lsid":
{"id":{"$binary":"P6RCGz9ZS4iWBSSHWXW15A==","$type":"4"},"uid":
{"$binary":"6Jo8PisnEi3dte03+pJFjdCyn/5cGQL8V2KqaoWsnk8=","$type":"0"}},"maxScan":18446744

// v2 formal for dml audit logs
```

```
{"atype":"authCheck","ts":1746473583711,"timestamp_utc":"2025-05-05
 19:33:03.711","remote_ip":"127.0.0.1:37754","users":
[{"user":"serviceadmin","db":"test"}],"param":
{"command":"find","ns":"test.col","args":{"find":"col","filter":{"x":1},"lsid":
{"id":{"$binary":"nJ88TGCSSd
+BeD2+ZtrhQg==","$type":"4"}},"$db":"test"},"result":0}}
```

- La condición del índice como parte del plan explicativo:

```
rs0:PRIMARY> db.col.createIndex({index1:1})
{
 "createdCollectionAutomatically" : false,
 "numIndexesBefore" : 1,
 "numIndexesAfter" : 2,
 "ok" : 1,
 "operationTime" : Timestamp(1761149251, 1)
}
```

La versión 2.0 del planificador explica el resultado del plan, mostrando la condición del índice y el 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)
}
```

Resultado del plan explicativo de la versión 1.0 del planificador:

```
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 versión 2.0 del planificador cierra la brecha de comportamiento con MongoDB

Hay algunas áreas en las que la versión 2.0 del planificador cierra las brechas de comportamiento con MongoDB:

- La versión 2.0 del planificador permite la búsqueda de índices numéricos en matrices aplanadas para `$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}}})
```

- Si bien la versión 1.0 del planificador excluía las cadenas en la proyección, el comportamiento de la versión 2.0 del planificador se alinea con MongoDB y las trata como valores literales.

```
// 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 versión 2.0 del planificador, como MongoDB, no permite la proyección en los mismos campos «x» y «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 versión 2.0 del planificador, como MongoDB, permite la proyección en subdocumentos:

```
// 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 versión 2.0 del planificador, como MongoDB, la proyección no admite campos después del operador $:

```
// 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 versión 2.0 del planificador, como MongoDB, permite el uso de `$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 }
```

Planificador de consultas v3

La versión 3 del planificador de Amazon DocumentDB 8.0 admite 21 etapas de agregación, incluidas 6 etapas nuevas. Planner V3 incluye soporte incorporado para distintos comandos. Ofrece hasta el doble de mejora del rendimiento general con respecto a Planner v2 en Amazon DocumentDB 5.0. Todas las funciones y operadores nuevos de Amazon DocumentDB 8.0 son compatibles con Planner v3. Entre las nuevas etapas de agregación de Amazon DocumentDB 8.0 compatibles con Planner v3 se incluyen $replaceWith, $VectorSearch, $merge, $set, $unset y $bucket. Planner v3 también admite nuevas funciones y operadores en Amazon DocumentDB 8.0, como Collation, Views, $merge, $pow, $rand, $dateTrunc, $date y $date. ToParts FromParts

Temas

- Requisitos previos
- Seleccionar la versión 3.0 del planificador como planificador de consultas predeterminado
- Prácticas recomendadas
- Limitaciones
- Mejoras en los operadores agregados y distintos
- Planifique los filtros de caché
- Posibles diferencias de comportamiento entre las versiones 1.0 y 3.0 del planificador y MongoDB

Requisitos previos

Los siguientes requisitos previos se aplican a la versión 3.0 del planificador:

- La versión 3.0 de Planner está disponible en todos los países en los Regiones de AWS que está disponible la versión 8.0 del motor.
- La versión 3.0 del planificador de consultas es el planificador de consultas predeterminado cuando se selecciona la versión 8.0 del motor.

Seleccionar la versión 3.0 del planificador como planificador de consultas predeterminado

Si cambió su planificador de consultas predeterminado en Amazon DocumentDB 8.0 y necesita volver al planificador v3, puede hacerlo desde la consola o la CLI:

- Siga los pasos en Modificación de parámetros de clúster de Amazon DocumentDB para modificar el grupo de parámetros del clúster.
- Para el parámetro titulado «plannerVersion», cambie el valor a 3.0 para indicar la versión 3.0 del planificador.
- Seleccione Aplicar inmediatamente (si selecciona Aplicar al reiniciar, la selección no será válida hasta el siguiente reinicio del clúster).

Prácticas recomendadas

Para obtener los resultados esperados, utilice las siguientes prácticas recomendadas al aplicar la versión 3.0 del planificador:

- En un clúster global, seleccione el mismo `plannerVersion` valor (1.0, 2.0 o 3.0) en los grupos de parámetros del clúster para ambas regiones. Ten en cuenta que seleccionar diferentes versiones del planificador en las regiones principales y secundarias puede provocar que el comportamiento y el rendimiento de las consultas sean inconsistentes.
- La actualización a la versión 3.0 del planificador durante un período de mantenimiento programado o durante períodos de tráfico reducido será lo menos molesto, ya que puede aumentar la tasa de errores si se cambia la versión del planificador cuando las cargas de trabajo se están ejecutando activamente.

Limitaciones

Las siguientes limitaciones se aplican a la versión 3.0 del planificador:

- La versión 3.0 de Planner no es compatible con los clústeres elásticos, que recurrirán a la versión 1.0 de Planner.

- Si bien Planner v1 permite el uso de «PlanHint» para garantizar que el optimizador de consultas seleccione un plan de consulta específico, Planner v3 no permite el uso de «PlanHint» y se basa en optimizaciones internas para elegir el mejor plan para una consulta determinada.

Mejoras en los operadores agregados y distintos

La versión 3.0 de Planner introduce mejoras en las etapas $aggregate y en el comando $distinct. Las siguientes son algunas de las mejoras más destacables.

- El planificador adelanta las etapas de $match siempre que es posible, lo que reduce la cantidad de documentos procesados en las etapas posteriores.

```
//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 } }
])
```

- El planificador combina automáticamente las etapas $lookup y $unwind cuando funcionan en el mismo campo, lo que reduce el procesamiento intermedio de datos y mejora el rendimiento.

```
                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 versión 3.0 de Planner presenta una nueva estrategia de ejecución de Distinct Scan que mejora significativamente el rendimiento de distintas operaciones en índices de cardinalidad bajos.

```
                 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"
                        }
                }
        }
}
```

Planifique los filtros de caché

La versión 3.0 de Planner admite los filtros de caché del plan, también denominados filtros de índice, que permiten restringir el conjunto de índices que el planificador considera para una forma de consulta específica. Los filtros se configuran con un comando de base de datos y se aplican en el servidor, de modo que, si se produce una regresión en las consultas, puede mitigarla sin modificar el código de la aplicación.

La versión 3.0 de Planner aplica filtros de caché de planes a más comandos que la versión 2.0 de Planner:

- El `distinct` comando requiere la versión 3.0 del planificador.
- El `aggregate` comando requiere la versión 3.0 del planificador y la 8.0.2 de Amazon DocumentDB. Se aplica un filtro cuando la canalización lee la colección a través de una fase inicial. `$match`

- Un filtro establecido en la colección extranjera de una `$graphLookup` etapa `$lookup` o etapa controla el índice utilizado para escanear esa colección extranjera. Esto también requiere la versión 3.0 del planificador y la 8.0.2 de Amazon DocumentDB.

Para obtener más información sobre la referencia de comandos, las formas de consulta compatibles, los ejemplos prácticos y la forma en que los filtros interactúan con las sugerencias, consulte. Planificar filtros de caché

Posibles diferencias de comportamiento entre las versiones 1.0 y 3.0 del planificador y MongoDB

En algunos casos extremos, es posible que la versión 3.0 del planificador produzca resultados que difieran ligeramente de los de la versión 1.0 del planificador. En esta sección se describen algunos ejemplos de estas posibilidades.

Feature Differences

- En una colección vacía o cuando las etapas anteriores, como Match, filtran todos los documentos, Planner v1 muestra el «campo» de salida :0. Planner v3 y MongoDB no ofrecerán ningún resultado.
- Con Plannerv1, {"$skip» :n} no omite si hay menos de n documentos devueltos en la etapa anterior. Plannerv3 y MongoDB omiten correctamente los documentos devueltos independientemente del número de documentos devueltos.
- Cuando no existe una colección externa a la que se hace referencia en $lookup, plannerv1 emite un error. Planner v3 y MongoDB tratan la colección externa como una colección vacía y ejecutan $lookup.

```
db.coll.aggregate([
    {$lookup: {from: "does_not_exist", localField: "a", foreignField: "a", as:
 "c"}}
])
```

- Solo la versión 1 de Planner permitirá usar varias búsquedas de $ en un proceso. Planner v2/v3 generará un error y MongoDB no lo admite.

```
VectorSearch = { "$search": { "vectorSearch": {
   "vector": [0.2, 0.5, 0.8],
   "path": "vectorEmbedding",
   "similarity": "cosine",
```

```
   "k": 2,
   "efSearch": 1
   }}}
db.coll.aggregate([VectorSearch, VectorSearch])
```

- Solo la versión 3 de Planner funciona cuando la etapa de VectorSearch no es la primera etapa. En este caso, MongoDB generará un error y Planner v1 no admite la etapa $VectorSearch.

```
VectorSearch = { {"$vectorSearch": {
    "queryVector": [0.2, 0.5, 0.8],
    "path": "vectorEmbedding",
    "similarity": "euclidean",
    "limit": 4,
    "numCandidates": 100} }

db.coll.aggregate([$match:{}, VectorSearch])
```

Consulta de datos geoespaciales con Amazon DocumentDB

En esta sección se explica cómo puede consultar datos geoespaciales con Amazon DocumentDB. Tras leer esta sección, podrá responder a cómo almacenar, consultar e indexar datos geoespaciales en Amazon DocumentDB.

Temas

- Descripción general de
- Indexación y almacenamiento de datos geoespaciales
- Consulta de datos geoespaciales
- Limitaciones

Descripción general de

Los casos de uso más comunes de la tecnología geoespacial incluyen el análisis de proximidad de sus datos. Por ejemplo, “buscar todos los aeropuertos en un radio de 50 millas de Seattle” o “encontrar los restaurantes más cercanos desde una ubicación determinada”. Amazon DocumentDB utiliza la especificación GeoJSON para representar datos geoespaciales. GeoJSON es una especificación de código abierto para las formas JSON-formatting de un espacio de coordenadas. Las coordenadas GeoJSON capturan tanto la longitud como la latitud y representan las posiciones en una esfera similar a la Tierra.

Indexación y almacenamiento de datos geoespaciales

Amazon DocumentDB utiliza el tipo GeoJSON de tipo “Punto” para almacenar datos geoespaciales. Cada documento (o subdocumento) de GeoJSON se compone generalmente de dos campos:

- tipo: la forma que se representa, que indica a Amazon DocumentDB cómo interpretar el campo “coordenadas”. En este momento, Amazon DocumentDB solo admite puntos
- coordenadas: un par de latitud y longitud representado como un objeto en una matriz: [longitud, latitud]

Amazon DocumentDB también utiliza índices de 2dsphere para indexar datos geoespaciales. Amazon DocumentDB admite puntos de indexación. Amazon DocumentDB admite consultas de proximidad con la indexación de 2dsphere.

Consideremos un escenario en el que está creando una aplicación para un servicio de entrega de alimentos. Desea almacenar los pares de latitud y longitud de los restaurantes en Amazon DocumentDB. Para ello, cree primero un índice en el campo geoespacial que contenga el par de latitud y longitud.

```
use restaurantsdb
db.usarestaurants.createIndex({location:"2dsphere"})
```

La salida de este comando tendrá un aspecto similar al siguiente:

```
{
 "createdCollectionAutomatically" : true,
 "numIndexesBefore" : 1,
 "numIndexesAfter" : 2,
 "ok" : 1
}
```

Una vez que haya creado un índice, puede empezar a insertar datos en su colección de 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
      ]
   }
});
```

Consulta de datos geoespaciales

Amazon DocumentDB admite consultas de proximidad, inclusión e intersección de datos geoespaciales. Un buen ejemplo de consulta de proximidad es buscar todos los puntos (todos los aeropuertos) que están a menos de una distancia determinada y a más de una distancia de otro punto (ciudad). Un buen ejemplo de consulta de inclusión es buscar todos los puntos (todos los aeropuertos) que se encuentran en un lugar específico area/polygon (estado de Nueva York). Un

buen ejemplo de consulta de intersección es buscar un polígono (estado) que se interseque con un punto (ciudad). Puede utilizar los siguientes operadores geoespaciales para obtener información a partir de sus datos.

- **`$nearSphere`**- `$nearSphere` es un operador de búsqueda que permite buscar puntos del más cercano al más lejano de un punto GeoJSON.
- **`$geoNear`**- `$geoNear` es un operador de agregación que permite calcular la distancia en metros desde un punto GeoJSON.
- **`$minDistance`**- `$minDistance` es un operador de búsqueda que se utiliza junto con `$nearSphere` o `$geoNear` para filtrar documentos que se encuentran al menos a la distancia mínima especificada desde el punto central.
- **`$maxDistance`**- `$maxDistance` es un operador de búsqueda que se utiliza junto con `$nearSphere` o `$geoNear` para filtrar documentos que se encuentran como máximo a la distancia máxima especificada desde el punto central.
- **`$geoWithin`**- `$geoWithin` es un operador de búsqueda que permite buscar documentos con datos geoespaciales que existan completamente dentro de una forma específica, como un polígono.
- **`$geoIntersects`**- `$geoIntersects` es un operador de búsqueda que permite buscar documentos cuyos datos geoespaciales se crucen con un objeto GeoJSON específico.

> **Note**
>
> `$geoNeary` `$nearSphere` requieren un índice de 2dsphere en el campo GeoJSON que utilice en la consulta de proximidad.

Ejemplo 1

En este ejemplo, aprenderá a buscar todos los restaurantes (puntos) ordenados por la distancia más cercana a una dirección (punto).

Para realizar una consulta de este tipo, puede `$geoNear` utilizar el cálculo de la distancia entre un conjunto de puntos y otro punto. También puede agregar el `distanceMultiplier` para medir la distancia en kilómetros.

```
db.usarestaurants.aggregate([
   {
```

```
      "$geoNear":{
         "near":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "spherical":true,
         "distanceField":"DistanceKilometers",
         "distanceMultiplier":0.001
      }
   }
])
```

El comando anterior devolvería los restaurantes ordenados por distancia (del más cercano al más lejano) desde el punto especificado. La salida de este comando tendrá un aspecto similar al siguiente.

```
{ "_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**
>
> numLas opciones `limit` y de la `$geoNear` fase, así como el límite predeterminado de 100 documentos, están en desuso en Amazon DocumentDB 8.0 (versión 4.0.5748 del parche de motor) y versiones posteriores. Los siguientes ejemplos utilizan `limit` y num se aplican únicamente a Amazon DocumentDB 8.0 (versión 4.0.4448 del parche del motor) y a versiones anteriores.

Para limitar el número de resultados de una consulta, utilice la opción `limit` o 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
      }
   }
])
```

Ejemplo 2

En este ejemplo, aprenderá a buscar todos los restaurantes (puntos) en un radio de 2 kilómetros de una dirección (punto) específica. Para realizar una consulta de este tipo, puede utilizar

`$nearSphere` con un `$minDistance` mínimo y un `$maxDistance` máximo desde un punto GeoJSON

```
db.usarestaurants.find({
   "location":{
      "$nearSphere":{
         "$geometry":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "$minDistance":1,
         "$maxDistance":2000
      }
   }
},
{
   "name":1
})
```

El comando anterior devolvería los restaurantes a una distancia máxima de 2 kilómetros del punto especificado. La salida de este comando tendrá un aspecto similar al siguiente.

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
```

Limitaciones

Amazon DocumentDB no admite la consulta ni la indexación de polígonos,,,, y. LineString MultiPoint MultiPolygon MultiLineString GeometryCollection

Índice parcial

Un índice parcial indexa los documentos de una colección que cumplen un criterio de filtro especificado. La característica de índice parcial se admite en los clústeres basados en instancias de Amazon DocumentDB 5.0.

Temas

- Creación de un índice parcial

- Operadores admitidos
- Consulta mediante un índice parcial
- Funcionalidades de indexación parcial
- Limitaciones del índice parcial

Creación de un índice parcial

Para crear un índice parcial, utilice el método `createIndex()` con la opción `partialFilterExpression`. Por ejemplo, la siguiente operación crea un índice compuesto único en la colección de pedidos que indexa los documentos que tienen un `OrderID` y cuyo campo `isDelivered` es true:

```
db.orders.createIndex(
  {"category": 1, "CustomerId": 1, "OrderId": 1},
  {"unique": true, "partialFilterExpression":
    {"$and": [
      {"OrderId": {"$exists": true}},
      {"isDelivered": {"$eq": false}}
    ]}
  }
)
```

Operadores admitidos

- $eq
- $exists
- $and (solo en el nivel superior)
- $gt/$gte/$lt/$lte (el escaneo de índices solo se usa cuando el filtro, basado en la consulta, coincide exactamente con la expresión del filtro parcial) (consulte las limitaciones)

Consulta mediante un índice parcial

Los siguientes patrones de consulta son posibles con índices parciales:

- El predicado de la consulta coincide exactamente con la expresión del filtro de índice parcial:

```
db.orders.find({"$and": [
```

```
    {"OrderId": {"$exists": true}},
    {"isDelivered": {"$eq": false}}
  ]}).explain()
```

- El resultado esperado del filtro de consulta es un subconjunto lógico del filtro parcial:

```
db.orders.find({"$and": [
    {"OrderId": {"$exists": true}},
    {"isDelivered": {"$eq": false}},
    {"OrderAmount": {"$eq": "5"}}
  ]}).explain()
```

- Se puede utilizar un subpredicado de la consulta junto con otros índices:

```
db.orders.createIndex({"anotherIndex":1})
db.orders.find({ "$or": [
      {"$and": [
        {"OrderId": {"$exists": true}},
        {"isDelivered": {"$eq": false}}
      ]},
      {"anotherIndex": {"$eq": 5}}
    ]
  }).explain()
```

> **Note**
>
> Un planificador de consultas puede optar por utilizar un escaneo de colecciones en lugar de un escaneo de índices si es eficiente hacerlo. Esto suele ocurrir en el caso de colecciones o consultas muy pequeñas que devuelven una gran parte de una colección.

Funcionalidades de indexación parcial

Lista de índices parciales

Enumere índices parciales con parciales FilterExpression mediante la `getIndex` operación. Por ejemplo, la operación `getIndex` emitida en las listas de índices parciales con los campos clave, nombre y partialfilterExpressions:

```
db.orders.getIndexes()
```

En este ejemplo, se devuelve la siguiente salida:

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

Expresión de filtro parcial múltiple en la misma clave:orden

Se pueden crear diferentes índices parciales para las mismas combinaciones de campos (clave:orden). Estos índices deben tener un nombre distinto.

```
db.orders.createIndex(
  {"OrderId":1},
  {
    name:"firstPartialIndex",
    partialFilterExpression:{"OrderId":{"$exists": true}}
  }
)
```

```
db.orders.createIndex(
  {"OrderId":1},
  {
    name:"secondPartialIndex",
    partialFilterExpression:{"OrderId":{"$gt": 1000}}
  }
)
```

Ejecute la operación `getIndexes` para enumerar todos los índices de la colección:

```
db.orders.getIndexes()
```

En estos ejemplos, se devuelve la siguiente salida:

```
[
  {
    "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**
>
> Los nombres de los índices deben ser diferentes y se deben eliminar solo por su nombre.

Índices con propiedades parciales y TTL

También puede crear índices con propiedades parciales y TTL mediante la especificación de las opciones `partialFilterExpression` y `expireAfterSeconds` durante la creación del índice. Esto le permite tener más control sobre qué documentos se eliminan ahora de una colección.

Por ejemplo, puede tener un índice TTL que identifique los documentos que se van a eliminar después de un periodo de tiempo determinado. Ahora puede establecer condiciones adicionales sobre cuándo eliminar documentos mediante la opción de índice parcial:

```
db.orders.createIndex(
    { "OrderTimestamp": 1 },
    {
        expireAfterSeconds: 3600 ,
        partialFilterExpression: { "isDelivered": { $eq: true } }
    }
)
```

En este ejemplo, se devuelve la siguiente salida:

```
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1,
        "operationTime" : Timestamp(1234567890, 1)
}
```

Ejecute la operación `getIndexes` para enumerar los índices de la colección:

```
db.orders.getIndexes()
[
    {
        "v" : 4,
        "key" : {
```

```
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.orders"
    }
```

En este ejemplo, se devuelve la siguiente salida:

```
[
    {
        "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
    }
]
```

Limitaciones del índice parcial

La característica de índice parcial tiene las siguientes limitaciones:

- Las consultas de desigualdad en Amazon DocumentDB solo utilizarán un índice parcial cuando el predicado del filtro de consultas coincida exactamente con el `partialFilterExpression` y sea del mismo tipo de datos.

> **Note**
>
> Ni siquiera `$hint` se puede usar para forzar el IXSCAN en el caso anterior.

En el siguiente ejemplo, `partialFilterExpression` solo se aplica a `field1`, pero no a `field2`:

```
db.orders.createIndex(
  {"OrderAmount": 1},
  {"partialFilterExpression": { OrderAmount : {"$gt" : 5}}}
)

db.orders.find({OrderAmount : {"$gt" : 5}}) // Will use partial index
db.orders.find({OrderAmount : {"$gt" : 6}}) // Will not use partial index
db.orders.find({OrderAmount : {"$gt" : Decimal128(5.00)}}) // Will not use partial
 index
```

- No se admite `partialFilterExpression` con los operadores de matriz. La siguiente operación generará un error:

```
db.orders.createIndex(
  {"CustomerId":1},
  {'partialFilterExpression': {'OrderId': {'$eq': [1000, 1001, 1002]}}}
)
```

- Los siguientes operadores no se admiten en un FilterExpression campo parcial:
  - `$all` (operador de matrices)
  - `$mod` (operador de matrices)
  - `$or`
  - `$xor`
  - `$not`
  - `$nor`
- El tipo de datos de la expresión de filtro y el filtro deben ser iguales.

Búsquedas de texto con Amazon DocumentDB

La función de búsqueda de texto completo nativa de Amazon DocumentDB (Text Index v1) le permite realizar búsquedas de texto en grandes conjuntos de datos textuales mediante índices de texto para fines especiales. En esta sección se describen las funcionalidades de la característica de índice de texto y se proporcionan los pasos para crear y utilizar índices de texto en Amazon DocumentDB. También se enumeran las limitaciones de la búsqueda de texto.

Temas

- Funcionalidades admitidas
- Uso del índice de texto de Amazon DocumentDB
- Diferencias con MongoDB
- Prácticas recomendadas y directrices
- Índice de texto V2
- Limitaciones

Funcionalidades admitidas

La búsqueda de texto de Amazon DocumentDB admite las siguientes funcionalidades compatibles con la API de MongoDB:

- Cree índices de texto en un solo campo.
- Cree índices de texto compuestos que incluyan más de un campo de texto.
- Haga búsquedas de una o varias palabras.
- Controle los resultados de la búsqueda mediante ponderaciones.
- Ordene los resultados de la búsqueda por puntuación.
- Usa el índice de texto en la canalización de agregación.
- Busque la frase exacta.

Uso del índice de texto de Amazon DocumentDB

Para crear un índice de texto en un campo que contenga datos de cadena, especifique la cadena “texto” como se muestra a continuación:

Índice de campo único:

```
db.test.createIndex({"comments": "text"})
```

Este índice admite consultas de búsqueda de texto en el campo de cadena de “comentarios” de la colección especificada.

Cree un índice de texto compuesto en más de un campo de cadena:

```
db.test.createIndex({"comments": "text", "title":"text"})
```

Este índice admite consultas de búsqueda de texto en los campos de cadena de “comentarios” y “título” de la colección especificada. Puede especificar hasta 30 campos al crear un índice de texto compuesto. Una vez creadas, las consultas de búsqueda de texto consultarán todos los campos indexados.

> **Note**
>
> Solo se permite un índice de texto en cada colección.

Visualización de un índice de texto en una colección de Amazon DocumentDB

Puede utilizarla `getIndexes()` en su colección para identificar y describir los índices, incluidos los índices de texto, como se muestra en el siguiente ejemplo:

```
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
   }
]
```

Una vez que haya creado un índice, empiece a insertar datos en su colección de 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"}])
```

Ejecución de consultas de búsqueda de texto

Ejecute una consulta de búsqueda de texto de una sola palabra

Necesitará utilizar los operadores `$text` y `$search` para hacer búsquedas de texto. El ejemplo siguiente devuelve todos los documentos en los que el campo indexado de texto contiene la cadena “manzana” o “manzana” en otros formatos, como “manzanas”:

```
db.test.find({$text: {$search: "apple"}})
```

Salida:

La salida de este comando tendrá un aspecto similar al siguiente:

```
{ "_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" }
```

Ejecución de una búsqueda de texto de varias palabras

También puede hacer búsquedas de texto de varias palabras en sus datos de Amazon DocumentDB. El siguiente comando devuelve documentos con un campo de texto indexado que contiene las palabras “manzana” o “tarta”:

```
db.test.find({$text: {$search: "apple pie"}})
```

Salida:

La salida de este comando tendrá un aspecto similar al siguiente:

```
{ "_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" }
```

Ejecución de una búsqueda de texto con frases de varias palabras

Para una búsqueda de frases de varias palabras, utilice este ejemplo:

```
db.test.find({$text: {$search: "\"apple pie\""}})
```

Salida:

El comando anterior devuelve documentos con un campo de texto indexado que contiene la frase exacta “tarta de manzana”. La salida de este comando tendrá un aspecto similar al siguiente:

```
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
```

Ejecución de una búsqueda de texto con filtros

También puede combinar la búsqueda de texto con otros operadores de consulta para filtrar los resultados en función de criterios adicionales:

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "interest"}}]})
```

Salida:

El comando anterior devuelve los documentos con un campo de texto indexado que contiene cualquier forma de “interés” y una “calificación_por_estrellas” igual a 5. La salida de este comando tendrá un aspecto similar al siguiente:

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
{ "_id" : 6, "star_rating" : 5, "comments" : "interested in couch for sale, year
 2022" }
```

Limitación del número de documentos devueltos en una búsqueda de texto

Puede optar por restringir el número de documentos devueltos mediante `limit`:

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "couch"}}]}).limit(1)
```

Salida:

El comando anterior devuelve un resultado que cumple con el filtro:

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
```

Ordenación de los resultados por puntuación de texto

En el siguiente ejemplo, se ordenan los resultados de la búsqueda de texto por puntuación de texto:

```
db.test.find({$text: {$search: "apple"}}, {score: {$meta: "textScore"}}).sort({score:
 {$meta: "textScore"}})
```

Salida:

El comando anterior devuelve los documentos con un campo de texto indexado que contiene la palabra “manzana” o “manzana” en otros formatos, como “manzanas”, y ordena el resultado en función de la relación del documento con el término de búsqueda. La salida de este comando tendrá un aspecto similar al siguiente:

```
{ "_id" : 1, "star_rating" : 4, "comments" : "apple is red", "score" :
 0.6079270860936958 }
{ "_id" : 3, "star_rating" : 3, "comments" : "apples, oranges - healthy fruit",
 "score" : 0.6079270860936958 }
```

```
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven",
 "score" : 0.6079270860936958 }
```

`$text` y `$search` también son compatibles con los comandos `aggregate`, `count`, `findAndModify`, `update` y `delete`.

Operadores de agregación

Canalización de agregación mediante **`$match`**

```
db.test.aggregate(
   [{ $match: { $text: { $search: "apple pie" } } }]
)
```

Salida:

El comando anterior devuelve los siguientes resultados:

```
{ "_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 combinación de otros operadores de agregación

```
db.test.aggregate(
   [
      { $match: { $text: { $search: "apple pie" } } },
      { $sort: { score: { $meta: "textScore" } } },
      { $project: { score: { $meta: "textScore" } } }
   ]
)
```

Salida:

El comando anterior devuelve los siguientes resultados:

```
{ "_id" : 4, "score" : 0.6079270860936958 }
{ "_id" : 1, "score" : 0.3039635430468479 }
{ "_id" : 2, "score" : 0.3039635430468479 }
{ "_id" : 3, "score" : 0.3039635430468479 }
```

Especifique varios campos al crear un índice de texto

Puede asignar ponderaciones a un máximo de tres campos del índice de texto compuesto. La ponderación predeterminada asignada a un campo en un índice de texto es uno (1). La ponderación es un parámetro opcional y debe estar en el rango de 1 a 100 000.

```
db.test.createIndex(
   {
     "firstname": "text",
     "lastname": "text",
     ...
   },
   {
     weights: {
       "firstname": 5,
       "lastname":10,
       ...
     },
     name: "name_text_index"
   }
 )
```

Diferencias con MongoDB

La característica de índice de texto de Amazon DocumentDB utiliza un índice invertido con un algoritmo de frecuencia de términos. Los índices de texto son dispersos de forma predeterminada. Debido a las diferencias en la lógica de análisis, los delimitadores de tokenización y otros, es posible que no se devuelva el mismo conjunto de resultados que MongoDB para el mismo conjunto de datos o forma de consulta.

Existen las siguientes diferencias adicionales entre el índice de texto de Amazon DocumentDB y MongoDB:

- No se admiten los índices compuestos que utilizan índices que no sean de texto.
- Los índices de texto de Amazon DocumentDB no distinguen entre mayúsculas y minúsculas.
- Solo se admite el idioma inglés con el índice de texto.
- No se admite la indexación de texto de campos matriciales (o de varias claves). Por ejemplo, no se podrá crear un índice de texto en “a” con el documento {“a”: [“manzana”, “tarta”]}.
- No se admite la indexación de texto con caracteres comodín.

- No se admiten índices de texto únicos.
- No se admite la exclusión de un término.

Prácticas recomendadas y directrices

- Para obtener un rendimiento óptimo en las consultas de búsqueda de texto que impliquen la clasificación por puntuaciones de texto, cree el índice de texto antes de cargar los datos.
- Los índices de texto requieren almacenamiento adicional para poder disponer de una copia interna optimizada de los datos indexados. Esto tiene implicaciones de costos adicionales.

Índice de texto V2

Amazon DocumentDB 8.0 presenta una nueva versión del índice de texto (V2) que cambia el analizador de búsqueda de texto subyacente para aumentar la compatibilidad con MongoDB.

Además de las funcionalidades que ofrecen los índices de texto de la versión 1, los índices de texto de la versión 2 también ofrecen la siguiente compatibilidad:

- Cuando es posible, el planificador mueve las etapas de $match a una fase más temprana del proceso, lo que reduce la cantidad de documentos procesados en las etapas posteriores.
- Se ha mejorado la tokenización de los caracteres especiales en los campos de texto, como correos electrónicos, direcciones URL y rutas de archivos. La V2 puede analizar y hacer coincidir los tokens individuales de estas cadenas, mientras que la 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" }
```

Limitaciones

La búsqueda de texto tiene las siguientes limitaciones en Amazon DocumentDB:

- Los índices de texto almacenan los lexemas y su información de posición. El tamaño combinado de todos los lexemas y su información de posición, en un solo documento, está limitado a 1 MB.

Índices

Los índices mejoran el rendimiento de las consultas al permitir a Amazon DocumentDB localizar rápidamente los documentos sin necesidad de escanear todos los documentos de una colección. Puede crear índices en los campos que su aplicación consulta con frecuencia para reducir la latencia y los costos. I/O

Para ver más temas relacionados con los índices, consulta:

- Administración de índices de Amazon DocumentDB— crear y crear índices
- Uso de índices— mejores prácticas de indexación
- Índices de solución de problemas— resolver los errores y la sobrecarga de creación de índices
- ¿Cómo analizo el uso de los índices e identifico los índices no utilizados?— encontrar índices no utilizados o faltantes

Temas

- Administración de índices
- Tipos de índices
- Propiedades de índices

Administración de índices

En esta sección se tratan temas de administración de índices para Amazon DocumentDB.

Temas

- Creación de índices
- Mantenimiento de los índices
- Prácticas recomendadas para trabajar con índices

Creación de índices

La compilación de índices en Amazon DocumentDB requiere tomar una serie de decisiones:

- ¿Con qué rapidez debe completarse?
- ¿Es posible que no se pueda acceder a la colección mientras se realiza la compilación?
- ¿Qué cantidad de la potencia informática de una instancia se puede asignar a la compilación?
- ¿Qué tipo de índice se debe crear?

Esta sección le ayuda a responder a estas preguntas y proporciona los comandos y ejemplos de supervisión para crear un índice de Amazon DocumentDB en su colección de clústeres basada en instancias.

Directrices

Las siguientes pautas incluyen los límites básicos y las desventajas de configuración a la hora de crear nuevos índices:

- Compatibilidad con la versión de Amazon DocumentDB: si bien la indexación de un solo proceso de trabajo se admite en todas las versiones de Amazon DocumentDB, la indexación de varios procesos de trabajo solo se admite en las versiones 4.0 y 5.0 de Amazon DocumentDB.
- Compensación de rendimiento: aumentar el número de procesos de trabajo en el proceso de creación del índice aumenta la utilización de la CPU y la E/S de lectura en la instancia principal de la base de datos Amazon DocumentDB. Los recursos necesarios para crear un índice nuevo no estarán disponibles para su carga de trabajo en ejecución.
- Clústeres elásticos: la indexación em paralelo no se admite en los clústeres elásticos de Amazon DocumentDB.
- Clústeres sin servidor: la indexación paralela en clústeres sin servidor requiere una capacidad mínima de al menos 32 DCU. Por debajo de este umbral, se acepta el `workers` parámetro, pero la creación del índice se ejecuta con un solo trabajador. La cantidad máxima de trabajadores

en paralelo aumenta en función de la capacidad mínima configurada: hay aproximadamente `min_DCU / 8` trabajadores disponibles. Por ejemplo, con 32 DCU puede usar hasta 4 trabajadores y con 64 DCU puede usar hasta 8 trabajadores.

> **Note**
>
> Es necesario reiniciar el clúster después de cambiar la configuración de capacidad mínima para que entre en vigor el nuevo límite de trabajadores en paralelo.

- Número máximo de trabajadores (basado en instancias): el número máximo de trabajadores que puede configurar depende del tamaño de la instancia principal en el clúster de base de datos. Es la mitad del número total de vCPU de la instancia principal del clúster de base de datos. Por ejemplo, puede ejecutar un máximo de 32 procesos de trabajo en una instancia db.r6g.16xlarge que tenga 64 vCPU.

> **Note**
>
> La creación de índices paralelos se admite en instancias de 2 veces más de tamaño.

- Número mínimo de procesos de trabajo: el número mínimo de procesos de trabajo que puede configurar es uno. La configuración predeterminada para la creación de índices en clústeres basados en instancias es de dos procesos de trabajo. Sin embargo, puedes reducir el número de trabajadores a uno mediante la opción «subprocesos de trabajo». Esto ejecutará el proceso con un solo proceso de trabajo.
- Compresión de índices: Amazon DocumentDB no admite la compresión de índices. Los tamaños de datos para índices pueden ser superiores que cuando utiliza otras opciones.
- Indexación de varias colecciones: la mitad de las vCPU de la instancia principal del clúster de base de datos se pueden usar para los procesos de trabajo configurados que realizan la creación de índices en varias colecciones.
- Tipos de índice: consulte Tipos de índices la lista de tipos de índices admitidos en Amazon DocumentDB.

Tipos de creación de índices

Los cuatro tipos de compilaciones de índices son:

- Primer plano: la creación del índice en primer plano bloquea todas las demás operaciones de base de datos de la colección en la que se creó el índice, hasta que se complete la creación del índice. La compilación en primer plano de Amazon DocumentDB consta de cinco etapas.
- Primer plano (único): las compilaciones de índices en primer plano de un solo documento (único) bloquean otras operaciones de la base de datos, como las compilaciones normales en primer plano. A diferencia de la compilación básica en primer plano, la compilación única utiliza una etapa adicional (clasificación de claves 2) para buscar claves duplicadas. La compilación en primer plano (única) consta de seis etapas.
- Antecedentes: la creación de índices en segundo plano permite que otras operaciones de la base de datos se ejecuten en primer plano mientras se crea el índice. La compilación de fondo de Amazon DocumentDB consta de ocho etapas.
- Antecedentes (únicos): las compilaciones de índices en segundo plano para un solo documento (único) permiten que otras operaciones de la base de datos se ejecuten en primer plano mientras se crea el índice. A diferencia de la compilación básica de fondo, la compilación única utiliza una etapa adicional (clasificación de claves 2) para buscar claves duplicadas. La compilación de fondo (única) se compone de nueve etapas.

El tipo de creación de índices predeterminado es en primer plano.

Para crear un índice en segundo plano, añada el parámetro «background»: true al crear el índice.

Introducción

Para iniciar la creación de índices en una colección, utilice el comando CreateIndexes. De forma predeterminada, el comando ejecutará dos elementos de trabajo paralelos, lo que aumentará dos veces la velocidad del proceso de creación del índice.

Por ejemplo, el siguiente proceso de comando muestra cómo crear un índice para el campo «nombre_usuario» de un documento y aumentar la velocidad del proceso de indexación a cuatro trabajadores:

1. Cree índices en segundo plano con dos trabajadores paralelos en el clúster:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "background":true }]})
```

2. Para optimizar la velocidad del proceso de creación de índices, puede especificar el número de trabajadores mediante la opción «subprocesos de trabajo» («trabajadores»:<number>) del comando CreateIndexes de db.RunCommand.

   Aumente la velocidad del proceso a cuatro procesos de trabajo paralelos:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "workers":4}]})
```

> **Note**
>
> Cuanto mayor sea el número de procesos de trabajo, más rápido avanzará la creación del índice. Sin embargo, cuanto mayor sea el número de procesos de trabajo, mayor será el aumento de la carga en las vCPU y la E/S de lectura de la instancia principal. Asegúrese de que el clúster esté lo suficientemente aprovisionado para soportar el aumento de la carga sin degradar otras cargas de trabajo.

Estado del progreso de indexación

El proceso de creación de índices funciona inicializando, escaneando las colecciones, clasificando las claves y, finalmente, insertándolas mediante un generador de índices. El proceso tiene hasta seis etapas cuando se ejecuta en primer plano y hasta nueve etapas cuando se ejecuta en segundo plano. Puede ver las métricas de estado, como el porcentaje de finalización, el número total de bloques de almacenamiento escaneados, las claves clasificadas y las claves insertadas, etapa por etapa.

Supervise el progreso del proceso de indexación mediante el comando db.currentOp () del shell de mongo. Una finalización del 100 % de la última etapa indica que todos los índices se han creado correctamente:

```
db.currentOp({"command.createIndexes": { $exists : true } })
```

> **Note**
>
> La visualización del estado del progreso de la indexación se admite solamente en Amazon DocumentDB 5.0.

Etapas de compilación del índice

Etapa	Primer plano	Primer plano (único)	Introducción	Fondo (único)
Inicializando	1	1	1	1
creando índice: inicializando	2	2	2	2.
creando índice: escaneando colección	3	3	3	3
creando índice: clasificando claves 1	4	4	4	4
creando índice: clasificando claves 2		5		5
creando índice: insertando claves	5	6	5	6
validando: escaneando índice			6	7
validando: clasificando tuplas			7	8

Etapa	Primer plano	Primer plano (único)	Introducción	Fondo (único)
validando: escaneando colección			8	9

- inicializando: createIndex está preparando el generador de índices. Esta fase debe ser muy breve.
- creando índice: inicializando: el generador de índices se está preparando para crear el índice. Esta fase debe ser muy breve.
- creando índice: escaneando colección: el generador de índices está escaneando la colección para recopilar las claves del índice. La unidad de medida son los «bloques».

> **Note**
>
> Si hay más de un elemento de trabajo configurado para la compilación del índice, se mostrará en esta etapa. La etapa de «recopilación por escaneo» es la única etapa en la que se utilizan varios trabajadores durante el proceso de creación del índice. En todas las demás etapas se mostrará un único proceso de trabajo.

- creando índice: clasificando claves 1: el generador de índices está clasificando las claves de índice recopiladas. La unidad de medida son las «claves».
- creando índice: clasificando claves 2: el generador de índices está clasificando las claves de índice recopiladas que corresponden a tuplas muertas. Esta fase solo existe para la compilación de índices únicos. La unidad de medida son las «llaves».
- creando índice: insertando claves: el generador de índices está insertando claves de índice en el nuevo índice. La unidad de medida son las «llaves».
- validando: escaneando índice: createIndex está escaneando el índice para buscar las claves que deben validarse. La unidad de medida son los «bloques».
- validando: clasificando tuplas: createIndex está ordenando el resultado de la fase de escaneo del índice.
- validando: escaneando colección: createIndex está escaneando la colección para validar las claves de índice encontradas en las dos fases anteriores. La unidad de medida son los «bloques».

Ejemplo de salida de compilación del índice

En el siguiente ejemplo de salida (creación del índice en primer plano), se muestra el estado de la creación del índice. El campo «msg» resume el progreso de la compilación indicando la fase y el porcentaje de finalización de la misma. El campo «trabajadores» indica la cantidad de trabajadores utilizados durante esa etapa de la creación del índice. El campo «progreso» muestra los números reales utilizados para calcular el porcentaje de finalización.

> **Note**
>
> Los campos IndexBuildName «actual», «mensaje» y «progreso» no son compatibles con la versión 4.0 de Amazon DocumentDB.

```
{
    "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
}
```

Recursos para la resolución de problemas

Consulte los siguientes recursos para solucionar problemas de creación de índices:

- La creación del índice falla
- Problemas de latencia en la creación del índice en segundo plano y errores

Mantenimiento de los índices

Sobrecarga del índice

Amazon DocumentDB usa el control de Multi-Version concurrencia (MVCC) para administrar las transacciones simultáneas. Cuando los documentos se eliminan o actualizan, sus versiones anteriores permanecen en las colecciones e índices como versiones «inactivas». El proceso de recopilación de elementos no utilizados recupera automáticamente el espacio de estas versiones inactivas para futuras operaciones.

La sobrecarga de índices se produce cuando el tamaño de los índices de una colección aumenta debido a la acumulación de índices inactivos u obsoletos o a la fragmentación de las páginas. El porcentaje informado representa la cantidad de espacio de índice que pueden utilizar las entradas de índice futuras. Esta sobrecarga consume espacio tanto en la memoria caché como en el almacenamiento del búfer. Si desea eliminar la sobrecarga, tendrá que recopilar los índices.

Ejemplo: ejecute el siguiente comando para determinar el almacenamiento no utilizado de su índice:

```
db.coll.aggregate({$indexStats:{}});
```

El resultado tendrá un aspecto parecido al siguiente:

```
{
    "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
    }
}
```

Puede reconstruir los índices sin tiempo de inactividad mediante el comando ReIndex, que requiere un análisis de toda la colección. Consulte Mantenimiento de índices mediante ReIndex.

Mantenimiento de índices mediante ReIndex

ReIndex es un comando que se usa para reconstruir un índice. Se utiliza normalmente cuando un índice está dañado o es ineficiente. Con el tiempo, los índices pueden acumular espacio no utilizado debido a muchas actualizaciones, inserciones o eliminaciones, lo que reduce el rendimiento. La reindexación ayuda a eliminar ese espacio no utilizado y a restablecer la eficiencia del índice.

Pautas para reindexar

- ReIndex solo es compatible con Amazon DocumentDB 5.0 y versiones posteriores
- ReIndex siempre se ejecuta en segundo plano.
- Amazon DocumentDB admite la reindexación de un único índice en segundo plano, lo que permite trabajar con varios trabajadores. Las consultas pueden utilizar el índice anterior cuando se ejecuta el proceso de reindexación.
- Amazon DocumentDB admite la indexación del informe de progreso a través de CurrentOp. Puede ver las etapas de creación del índice de forma similar a las etapas de creación del índice vistas

durante la creación del índice. La única diferencia es que ReIndex siempre tiene ocho etapas, independientemente de si es único o no. No existe la etapa «crear un índice: clasificar las claves 2".

- ReIndex se puede ejecutar simultáneamente con cualquier comando, excepto los comandos relacionados con índices de la misma colección: CreateIndexes, DropIndexes, CollMod y RenameCollection.
- Actualmente, ReIndex no es compatible con los índices de texto, geoespaciales y vectoriales.
- La compatibilidad con ReIndex para índices parciales está disponible en la versión 8.0.2 y posteriores del motor.

Compilación de ReIndex

Use el siguiente comando para reconstruir el índice:

```
db.runCommand({ reIndex: "collection-name", index: "index-name"})
```

Si lo desea, también puede controlar el número de procesos de trabajo asignados al proceso de reconstrucción:

```
db.runCommand({ reIndex: "collection-name", index: "index-name", workers: number })
```

Para obtener información específica sobre la administración de índices con Java, consulte. Administración de índices en Amazon DocumentDB con Java

Prácticas recomendadas para trabajar con índices

Creación de índices

Al importar datos en Amazon DocumentDB, debe crear los índices antes de importar grandes conjuntos de datos. Puede usar la herramienta de índice de Amazon DocumentDB para extraer índices de una instancia de MongoDB o un directorio de mongodump en ejecución y crear esos índices en un clúster de Amazon DocumentDB. Para obtener más información acerca de las migraciones, consulte Migración y actualización.

Selectividad de índice

Limite la creación de índices a los campos en los que el número de valores duplicados sea inferior al 1% del número total de documentos de la colección. Por ejemplo, si su colección contiene 100 000 documentos, cree índices únicamente en los campos en los que el mismo valor aparezca 1000 veces o menos.

La elección de un índice con un número elevado de valores únicos (es decir, una cardinalidad alta) garantiza que las operaciones de filtrado arrojen un número reducido de documentos, lo que permite obtener un buen rendimiento durante los escaneos del índice. Un ejemplo de un índice de alta cardinalidad es un índice único, que garantiza que los predicados de igualdad devuelvan como máximo un documento único. Entre los ejemplos de baja cardinalidad se incluyen un índice sobre un campo booleano y un índice sobre el día de la semana. Debido a su bajo rendimiento, es poco probable que el optimizador de consultas de la base de datos elija índices de cardinalidad bajos. Al mismo tiempo, los índices de cardinalidad bajos siguen consumiendo recursos como espacio en disco y. I/Os Como regla general, debe aplicar los índices a campos donde la frecuencia del valor típica sea un 1 % del tamaño total de la colección o menos.

Además, se recomienda crear índices únicamente en campos que se utilizan comúnmente como filtro y buscar regularmente índices no utilizados. Para obtener más información, consulte the section called “Análisis del uso de los índices e identificar los índices no utilizados”.

Impacto de los índices en los datos de escritura

Aunque los índices pueden mejorar el rendimiento de las consultas al evitar la necesidad de examinar todos los documentos de una colección, esta mejora tiene algún inconveniente. Para cada índice de una colección, cada vez que se inserta, actualiza o elimina un documento, la base de datos debe actualizar la colección y escribir los campos en cada uno de los índices de la colección. Por ejemplo, si una colección tiene nueve índices, la base de datos debe realizar diez escrituras antes de validar la operación para el cliente. Por lo tanto, cada índice adicional implica una latencia de escritura adicional y un aumento en el almacenamiento utilizado en general. I/Os

Las instancias del clúster deben tener el tamaño adecuado para mantener toda la memoria del conjunto de trabajo. Esto evita la necesidad de leer continuamente las páginas de índice del volumen de almacenamiento, lo que repercute negativamente en el rendimiento y genera I/O costos más altos. Para obtener más información, consulte Determinación del tamaño de las instancias.

Para obtener un mejor rendimiento, reduzca el número de índices de sus colecciones, agregando solo los índices necesarios para mejorar el rendimiento de las consultas comunes. Aunque las

cargas de trabajo varían, una buena directriz es mantener el número de índices por colección en cinco o menos.

Identificación de índices que faltan

Como práctica recomendada, identifique los índices que faltan de forma regular. Para obtener más información, consulte the section called “Identificación de los índices omitidos”.

Identificación de índices no utilizados

Como práctica recomendada, identifique y elimine los índices no utilizados de forma regular. Para obtener más información, consulte the section called “Análisis del uso de los índices e identificar los índices no utilizados”.

Tipos de índices

Temas

- Índices de campo único
- Índices compuestos
- Multi-key índices
- Índices de texto
- Indexación geoespacial
- Índices vectoriales

Índices de campo único

Los índices de campo único almacenan información de cualquier campo individual de una colección de documentos. De forma predeterminada, todas las colecciones tienen un índice en el `_id` campo. Puede agregar índices adicionales para acelerar las consultas y operaciones importantes.

Los índices de campo único son beneficiosos cuando:

- Su aplicación consulta o filtra con frecuencia los documentos en función de los valores de un campo específico.
- Debe ordenar los documentos de manera eficiente por un campo en particular.
- Desea garantizar la unicidad de un campo específico en todos los documentos de una colección mediante la creación de un índice único.

Propiedades de índice compatibles

Opción	3.6	4.0	5.0	8.0	Elastic Cluster
name	Sí	Sí	Sí	Sí	Sí
único	Sí	Sí	Sí	Sí	Sí
escaso *	Sí	Sí	Sí	Sí	Sí
parcial * FilterExpression	No	No	Sí	Sí	No
caducarAfterSeconds	Sí	Sí	Sí	Sí	Sí

* Las `partialFilterExpression` opciones `sparse` y no se pueden usar juntas en la misma definición de índice. Si intenta crear un índice con estas opciones, se producirá el siguiente error:

```
Error in specification: cannot mix partialFilterExpression and sparse options
```

Crear un índice de campo único

Utilice el `createIndex()` método para crear un índice de campo único. La sintaxis es la siguiente: `db.collection.createIndex(<key>, <options>)`.

El `key` parámetro es un documento JSON que especifica el orden de clasificación del campo y del índice:

```
{
  "<field>": <1 (ascending)|-1 (descending)>
}
```

El `options` parámetro es un documento JSON que especifica las opciones del índice:

```
{
```

```
  "name": "<name>",
  "unique": <true | false>,
  "sparse": <true | false>,
  "partialFilterExpression": <filter expression>,
  "expireAfterSeconds": <seconds before expiry>
}
```

Consulte Propiedades del índice para ver ejemplos de cómo crear índices de campo único.

Índices compuestos

Los índices compuestos almacenan información de dos o más campos de una colección de documentos, lo que permite realizar consultas en el primer campo o en cualquier campo de prefijo. Estos índices optimizan el rendimiento de las consultas que filtran varios campos simultáneamente o combinan el filtrado con las operaciones de clasificación. También son eficaces para las consultas con una sola condición en los campos indexados situados más a la izquierda. La base de datos aprovecha estas entradas de índice para localizar de manera eficiente los documentos coincidentes sin necesidad de escanear toda la colección.

Los índices de campos compuestos son beneficiosos cuando:

- Es necesario filtrar varios campos de forma simultánea.
- Debe combinar el filtrado con las operaciones de clasificación.

Propiedades de índice compatibles

Opción	3.6	4.0	5.0	8.0	Elastic Cluster
name	Sí	Sí	Sí	Sí	Sí
único	Sí	Sí	Sí	Sí	Sí
escaso *	Sí	Sí	Sí	Sí	Sí
parcial * FilterExpression	No	No	Sí	Sí	No

* Las `partialFilterExpression` opciones `sparse` y no se pueden usar juntas en la misma definición de índice. Si intenta crear un índice con estas opciones, fallará y aparecerá el siguiente error:

```
Error in specification: cannot mix partialFilterExpression and sparse options
```

Creación de un índice compuesto

Utilice el `createIndex()` método para crear un índice compuesto. La sintaxis del método es: `db.collection.createIndex(<keys>, <options>)`

El `keys` parámetro es un documento JSON que especifica los campos y el orden de clasificación del índice:

```
{
  "<field 1>": <1 (ascending)|-1 (descending)>,
  "<field 2>": <1 (ascending)|-1 (descending)>,
  ...
}
```

Tenga en cuenta que solo un campo puede ser una matriz en un índice compuesto. Si intenta crear un índice compuesto en dos o más campos de matriz, se producirá el error siguiente:

```
multiple fields of compound index cannot be arrays
```

El `options` parámetro es un documento JSON que especifica las opciones del índice:

```
{
  "name": "<name>",
  "unique": <true | false>,
  "sparse": <true | false>,
  "partialFilterExpression": <filter expression>
}
```

Consulta las propiedades del índice para ver ejemplos de cómo crear índices compuestos.

Multi-key índices

En el caso de los campos que tienen un valor de matriz, un índice de varias claves permite crear una clave de índice para cada elemento de la matriz. Al indexar una matriz, se crea una entrada de índice para cada elemento de la matriz.

Multi-key los índices son beneficiosos cuando la aplicación consulta o filtra con frecuencia documentos en función de los valores de las matrices.

Propiedades de índice compatibles

Opción	3.6	4.0	5.0	8.0	Elastic Cluster
name	Sí	Sí	Sí	Sí	Sí
único	Sí	Sí	Sí	Sí	Sí
escaso *	Sí	Sí	Sí	Sí	Sí
parcial * FilterExpression	No	No	Sí	Sí	No
caducarAfterSeconds	Sí	Sí	Sí	Sí	Sí

* Las `partialFilterExpression` opciones `sparse` y no se pueden usar juntas en la misma definición de índice. Si intenta crear un índice con estas opciones, fallará y aparecerá el siguiente error:

```
Error in specification: cannot mix partialFilterExpression and sparse options
```

Crear un índice de varias claves

Utilice el `createIndex()` método para crear un índice de varias claves. La sintaxis del método es: `db.collection.createIndex(<key>, <options>)`

El `key` parámetro es un documento JSON que especifica el orden de clasificación del campo y del índice:

```
{
  "<field>": <1 (ascending)|-1 (descending)>
}
```

El `options` parámetro es un documento JSON que especifica las opciones del índice:

```
{
  "name": "<name>",
  "unique": <true | false>,
  "sparse": <true | false>,
  "partialFilterExpression": <filter expression>,
  "expireAfterSeconds": <seconds before expiry>
}
```

En el siguiente ejemplo, se crea un índice de varias claves en el `categories` campo, ordenado en orden ascendente con el nombre: `book_categories`

```
db.collection.createIndex(
  {
    "categories": 1
  },
  {
    "name": "book_categories"
  }
)
```

Consulte Propiedades del índice para ver ejemplos de cómo crear índices de varias claves.

Índices de texto

Los índices de texto son útiles para buscar palabras o frases en los campos de texto de los documentos, ya que permiten realizar la función de búsqueda de texto completo. Puede crear índices de texto de campo único o índices de texto compuesto con más de un campo de texto. Sin embargo, solo puede tener un índice de texto por colección. Para obtener más información, consulte Búsquedas de texto con Amazon DocumentDB.

Propiedades de índice compatibles

Opción	3.6	4.0	5.0	8.0	Elastic Cluster
name	No	No	Sí	Sí	No

Creación de un índice de texto

Utilice el `createIndex()` método para crear un índice de texto. La sintaxis del método es: `db.collection.createIndex(<keys>, <options>)`

El `keys` parámetro es un documento JSON que especifica los campos y el tipo de índice de texto:

```
{
  "<field 1>": "text",
  "<field 2>": "text"
}
```

El `options` parámetro es un documento JSON que especifica el nombre del índice:

```
{
  "name": "<name>"
}
```

Consulte Propiedades del índice para ver ejemplos de cómo crear índices de texto.

Indexación geoespacial

Los índices geoespaciales son un tipo de índice especializado diseñado para consultar y administrar de manera eficiente los datos geoespaciales almacenados en una colección de documentos. Amazon DocumentDB admite los índices de 2dsphere, que están diseñados específicamente para gestionar los datos geoespaciales de una esfera (como la Tierra). Esto permite realizar cálculos y consultas precisos basados en la geometría esférica.

Los índices geoespaciales son beneficiosos cuando las aplicaciones necesitan realizar consultas basadas en la ubicación, como:

- buscar puntos de interés cercanos,
- determinar si una ubicación se encuentra dentro de un área específica
- calcular distancias entre ubicaciones

Propiedades de índice compatibles

Opción	3.6	4.0	5.0	8.0	Elastic Cluster
name	Sí	Sí	Sí	Sí	Sí

Creación de un índice geoespacial

Utilice el `createIndex()` método para crear un índice geoespacial. La sintaxis del método es: `db.collection.createIndex(<key>, <options>)`

El `key` parámetro es un documento JSON que especifica el campo y el tipo de índice de 2dsphere:

```
{
  "<field>": "2dsphere"
}
```

El `options` parámetro es un documento JSON que especifica las opciones del índice:

```
{
  "name": "<name>"
}
```

Consulte Propiedades del índice para ver ejemplos de cómo crear índices geoespaciales.

Índices vectoriales

Los índices vectoriales son un tipo de índice especializado diseñado para consultar y administrar de manera eficiente los datos vectoriales almacenados en una colección de documentos. Amazon DocumentDB admite los índices Hierarchical Navigable Small World (HNSW) e Inverted File with Flat Compression (IVFflat).

Para obtener más información, consulte Búsqueda vectorial para Amazon DocumentDB.

Los índices vectoriales son beneficiosos para los casos de uso del aprendizaje automático y la IA generativa, como:

- búsqueda semántica
- recomendación de producto

- personalización
- chatbots
- detección de fraude
- detección de anomalías

Propiedades de índice compatibles

Opción	3.6	4.0	5.0	8.0	Elastic Cluster
name	No	No	Sí	Sí	No

Creación de un índice vectorial

Utilice el comando CreateIndex con el `runCommand()` método para crear un índice vectorial. La sintaxis es la siguiente:

```
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]
    }
  }]
})
```

El parámetro clave es un documento JSON que especifica el tipo de índice vectorial y de campo:

```
{
```

```
  "<field>": "vector"
}
```

Consulte Propiedades del índice para ver ejemplos de cómo crear índices vectoriales.

Propiedades de índices

Las propiedades de los índices definen el comportamiento y las características de los índices en Amazon DocumentDB. Comprender estas propiedades le ayuda a diseñar índices eficientes que cumplan con los requisitos específicos de su aplicación.

Temas

- Propiedad del índice: nombre
- Propiedad de índice: única
- Propiedad de índice: dispersa
- Propiedad del índice: parcial FilterExpression
- Propiedad del índice: caducar AfterSeconds
- Propiedad de índice: oculta

Propiedad del índice: nombre

Tipos de índices compatibles

Opción	3.6	4.0	5.0	8.0	Clúster elástico
campo único	Sí	Sí	Sí	Sí	Sí
compuesto	Sí	Sí	Sí	Sí	Sí
tecla múltiple	Sí	Sí	Sí	Sí	Sí
text	No	No	Sí	Sí	No
geoespacial	Sí	Sí	Sí	Sí	Sí
vector	No	No	Sí	Sí	No

Utilice la opción de nombre para proporcionar un nombre opcional para el índice.

Todos los ejemplos utilizan el siguiente documento de ejemplo:

```
{
  "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 único

```
db.collection.createIndex(
  {
    "productId": 1
  },
  {
    "name": "single_field_index"
  }
)
```

Compuesta

```
db.collection.createIndex(
  {
    "productId": 1,
    "manufacturer": 1
  },
  {
    "name": "compound_index"
  }
)
```

Multi-key

```
db.collection.createIndex(
  {
    "tags": 1
  },
  {
    "name": "multikey_index"
  }
)
```

Texto

```
db.collection.createIndex(
  {
    "name": "text"
  },
  {
    "name": "text_index"
  }
)
```

Geospatial (Geoespacial)

```
db.collection.createIndex(
  {
    "supplier.location": "2dsphere"
  },
  {
    "name": "geospatial_index"
```

```
  }
)
```

Vector

```
db.runCommand({
  "createIndexes": "collection",
  "indexes": [{
    "key": {
      "productEmbedding": "vector"
    },
    "name": "hnsw_index",
    "vectorOptions": {
      "type": "hnsw",
      "dimensions": 2,
      "similarity": "euclidean",
      "m": 16,
      "efConstruction": 64
    }
  }]
})
```

Propiedad de índice: única

Tipos de índices compatibles

Opción	3.6	4.0	5.0	8.0	Clúster elástico
campo único	Sí	Sí	Sí	Sí	Sí
compuesto	Sí	Sí	Sí	Sí	Sí
tecla múltiple	Sí	Sí	Sí	Sí	Sí

Utilice la opción exclusiva para garantizar la unicidad de los campos en todos los documentos de una colección.

Ejemplos

Los siguientes ejemplos muestran cómo crear índices únicos en el siguiente documento de ejemplo:

```
{
  "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 único

Crea un índice de campo único en el ProductID para asegurarte de que el mismo ProductID no exista en más de un documento:

```
db.collection.createIndex(
  {
    "productId": 1
  },
  {
    name: "productId_unique",
    unique: true
  }
```

```
)
```

Compuesto

Crea un índice compuesto único sobre el SKU y el fabricante para asegurarte de que la misma combinación de SKU y fabricante no exista en más de un documento:

```
db.collection.createIndex(
  {
    "sku": 1,
    "manufacturer": 1
  },
  {
    name: "sku_and_manufacturer_unique",
    unique: true
  }
)
```

clave múltiple

Cree un índice único de códigos de barras con múltiples claves para asegurarse de que ningún valor de la matriz de códigos de barras exista en más de un documento:

```
db.collection.createIndex(
  {
    "barcodes": 1
  },
  {
    name: "barcodes_unique",
    unique: true
  }
)
```

Al indexar una matriz, se crea una entrada de índice para cada elemento de la matriz. Por ejemplo, si una matriz tiene 50 elementos, tiene 50 entradas de índice. Por este motivo, los índices multiclave únicos garantizan la unicidad de todos los elementos individuales. Por ejemplo, los siguientes documentos infringirán la restricción de exclusividad en el índice de campos de matriz de valores:

```
{ "values": [ 1, 2, 3] }
{ "values": [ 3, 2 ] }   --> 3 and 2 already exist
```

```
{ "values": [ 1 ] }      --> 1 already exists
```

Tenga en cuenta el siguiente comportamiento con índices únicos:

1. Si crea un índice único a partir de datos existentes en el que dos (o más) documentos tienen los mismos valores para los campos indizados, la creación del índice fallará y aparecerá el siguiente error: `could not create unique index: <collection> index: <index name>`
2. Si inserta un documento en el que el valor del campo indizado coincide con el valor de ese campo en otro documento, la inserción no se realizará correctamente y aparecerá el siguiente error: `E11000 duplicate key error collection: <collection> index: <index name>`
3. Si actualizas un documento existente para que el nuevo valor del campo indexado coincida con el valor de ese campo en otro documento, la actualización fallará y aparecerá el siguiente error: `E11000 duplicate key error collection: <collection> index: <index name>`
4. Si el campo indexado no aparece en un documento, el valor se considerará nulo. La creación, inserción y actualización del índice fallarán tal y como se describió anteriormente si el campo indexado no aparece en dos (o más) documentos.

Propiedad de índice: dispersa

Tipos de índices compatibles

Opción	3.6	4.0	5.0	8.0	Clúster elástico
campo único	Sí	Sí	Sí	Sí	Sí
compuesto	Sí	Sí	Sí	Sí	Sí
tecla múltiple	Sí	Sí	Sí	Sí	Sí

Utilice la opción dispersa para omitir la indexación de documentos en los que no figuran los campos indexados, lo que reduce el tamaño del índice y ahorra espacio en la memoria. Como el tamaño del índice es más pequeño, las consultas que lo utilizan son más eficaces. Para que una consulta utilice un índice disperso, debe utilizar la cláusula $exists en los campos indizados. Si omite la cláusula $exists, Amazon DocumentDB no utilizará el índice disperso.

Ejemplos

Los siguientes ejemplos muestran cómo crear índices dispersos en el siguiente documento de ejemplo:

```
{
  "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')
}
```

Tenga en cuenta que los campos de reseñas, material y color no existen en todos los documentos.

campo único

Cree un índice de campo único disperso en el campo de material:

```
db.collection.createIndex(
  {
    "material": 1
```

```
  },
  {
    "name": "material_sparse",
    "sparse": true
  }
)
```

Este índice se utilizará para buscar todos los productos que tengan un valor de material:

```
db.collection.find({
  "material": {
    $exists: true
  }
})
```

compuesto

Cree un índice compuesto disperso en los campos de material y color:

```
db.collection.createIndex(
  {
    "material": 1,
    "color": 1
  },
  {
    "name": "material_and_color_sparse",
    "sparse": true
  }
)
```

Este índice se utilizará para buscar todos los productos que tengan cualquier combinación de valores de material y color:

```
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
      }
    }
  ]
})
```

tecla múltiple

Cree un índice disperso de varias claves en la matriz de reseñas:

```
db.collection.createIndex(
  {
    "reviews": 1
  },
  {
    "name": "reviews_sparse",
    "sparse": true
```

```
  }
)
```

Este índice se utilizará para buscar todos los productos que tengan una matriz de reseñas:

```
db.collection.find({
  "reviews": {
    $exists: true
  }
})
```

Propiedad del índice: parcial FilterExpression

Tipos de índices compatibles

Opción	3.6	4.0	5.0	8.0	Clúster elástico
campo único	No	No	Sí	Sí	No
compuesto	No	No	Sí	Sí	No
tecla múltiple	No	No	Sí	Sí	No
geoespacial	No	No	No	No	No

Utilice la FilterExpression opción parcial para crear un índice parcial que solo incluya los documentos que cumplan una condición de filtro especificada. Esto permite crear índices más eficaces al indexar solo un subconjunto de documentos de una colección, en lugar de indexar todos los documentos, lo que reduce el tamaño del índice y ahorra espacio en la memoria. Como el tamaño del índice es más pequeño, las consultas que lo utilizan son más eficaces. Amazon DocumentDB utilizará el índice parcial en los siguientes escenarios:

- El predicado de la consulta coincide exactamente con la expresión del filtro de índice parcial.
- El resultado esperado del filtro de consulta es un subconjunto lógico del filtro parcial.
- Un subpredicado de la consulta se puede usar junto con otros índices.

Para obtener más información, consulte Índice parcial.

Ejemplos

En los siguientes ejemplos se muestra cómo crear índices parciales en el siguiente documento de ejemplo:

```
{
  "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 único

Crea un índice parcial de campo único sobre el fabricante para los productos de color gris espacial:

```
db.collection.createIndex(
  {
    "manufacturer": 1
  },
  {
```

```
    "name": "manufacturer_space_gray",
    "partialFilterExpression": {
      "color": {
        $eq: "Space Gray"
      }
    }
  }
)
```

Este índice se utilizará para buscar productos de color gris espacial por fabricante:

```
db.collection.find({
  "manufacturer": "The Manufacturer",
  "color": {
    $eq: "Space Gray"
  }
})
```

compuesto

Cree un índice compuesto parcial sobre el fabricante y el color de los productos que tienen un material de policarbonato:

```
db.collection.createIndex(
  {
    "manufacturer": 1,
    "color": 1
  },
  {
    "name": "manufacturer_and_color_polycarbonate",
    "partialFilterExpression": {
      "material": {
        $eq: "Polycarbonate"
      }
    }
  }
)
```

Este índice se utilizará para buscar productos de policarbonato por fabricante y color:

```
db.collection.find({
```

```
  "manufacturer": "The Manufacturer",
  "color": "Space Gray",
  "material": {
    $eq: "Polycarbonate"
  }
})
```

tecla múltiple

Crea un índice parcial con varias claves en las etiquetas de los productos cuyo fabricante sea The Manufacturer:

```
db.collection.createIndex(
  {
    "tags": 1
  },
  {
    "name": "tags_the_manufacturer",
    "partialFilterExpression": {
      "manufacturer": {
        $eq: "The Manufacturer"
      }
    }
  }
)
```

Este índice se utilizará para buscar productos etiquetados con una impresora cuyo fabricante sea El fabricante:

```
db.collection.find({
  "tags": "printer",
  "manufacturer": {
    $eq: "The Manufacturer"
  }
})
```

Propiedad del índice: caducar AfterSeconds

Tipos de índices compatibles

Tipo de índice	3.6	4.0	5.0	8.0	Clúster elástico
campo único	Sí	Sí	Sí	Sí	Sí
tecla múltiple	Sí	Sí	Sí	Sí	Sí

Utilice la AfterSeconds opción de caducidad para crear un índice de tiempo de vida (TTL). Los índices TTL permiten eliminar documentos en función de su antigüedad en función del tiempo de espera de cada documento. Cuando un documento alcanza la antigüedad TTL especificada, se elimina de la colección.

Para conocer las prácticas recomendadas en relación con las eliminaciones mediante TTL, consulte. TTL y cargas de trabajo de series temporales

Ejemplos

En los siguientes ejemplos se muestra cómo crear índices TTL en el siguiente documento de ejemplo:

```
{
  "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 único

Cree un índice TTL en LastUpdated para eliminar los documentos que no se hayan actualizado en 90 días:

```
db.collection.createIndex(
  {
    "lastUpdated": 1
  },
  {
    "name": "lastUpdated_ttl",
    "expireAfterSeconds": 7776000
  }
)
```

clave múltiple

Crea un índice TTL en reviews.review_date para eliminar los documentos que no hayan sido revisados durante el último año:

```
db.collection.createIndex(
  {
    "reviews.review_date": 1
  },
  {
    "name": "reviews_review_date_ttl",
    "expireAfterSeconds": 31536000
  }
)
```

Tenga en cuenta que si el índice TTL está en un campo de matriz, se comprueban todos los elementos de la matriz. Si alguna marca de tiempo de la matriz cumple la condición de tiempo de espera, se eliminará el documento.

Propiedad de índice: oculta

Amazon DocumentDB admite la propiedad de `hidden` índice a partir de Amazon DocumentDB 8.0.1+ y Amazon DocumentDB 5.0.2+.

Los índices ocultos no son visibles para el planificador de consultas y no se pueden usar para respaldar una consulta. Si oculta un índice del planificador, puede evaluar el posible impacto de eliminar el índice sin eliminarlo realmente. Si el impacto es negativo, puedes mostrar el índice en lugar de tener que volver a crear un índice perdido.

La `hidden` propiedad es compatible con todos los tipos de índice, incluidos los índices de campo único, compuestos, multiclave, de texto, geoespaciales y vectoriales.

Comportamiento

Además de estar ocultos para el planificador, los índices ocultos se mantienen y aplican por completo. Por ejemplo:

- Si un índice oculto es un índice único, el índice sigue aplicando su restricción única a los documentos.
- Si un índice oculto es un índice TTL, los documentos seguirán caducando.
- Los índices ocultos se incluyen en `listIndexes` los resultados. `db.collection.getIndexes()`
- Los índices ocultos se actualizan al escribir operaciones en la colección y siguen consumiendo espacio en disco y memoria. Como resultado, se incluyen en operaciones estadísticas como `db.collection.stats()` y`$indexStats`.

- Al ocultar un índice oculto o mostrar un índice oculto, se restablece su índice. `$indexStats` Al ocultar un índice ya oculto o mostrar un índice que ya no está oculto, no se restablece. `$indexStats`

Restricciones

- No puede ocultar el `_id` índice.
- No se puede usar `cursor.hint()` para forzar el uso de un índice oculto.
- `$merge`falla si el índice único requerido en los on campos está oculto. Haga que el índice esté visible antes de ejecutarlo`$merge`.

Ejemplos

Crea un índice oculto

Para crear un índice en estado oculto, utilice el `db.collection.createIndex()` método y defina la `hidden` opción `true` en la definición del índice.

Por ejemplo, la operación siguiente crea un índice ascendente oculto en el `productId` campo:

```
db.collection.createIndex(
  { "productId": 1 },
  { "name": "productId_1", "hidden": true }
)
```

Para verificar, ejecute `db.collection.getIndexes()` o `listIndexes` en la colección. Un índice oculto incluye `"hidden": true` en su definición.

```
db.collection.getIndexes()
[
  {
    "v": 4,
    "key": { "_id": 1 },
    "name": "_id_"
  },
  {
    "v": 4,
    "key": { "productId": 1 },
    "name": "productId_1",
    "hidden": true
```

```
  }
]
```

La opción de índice solo `hidden` se devuelve si el valor es`true`.

Ocultar un índice existente

Utilice el `collMod` comando para ocultar un índice existente. Identifique el índice por su `name` o por su valor y `keyPattern` establézcalo `hidden` en`true`.

El ejemplo siguiente oculta un índice por su nombre.

```
db.runCommand({
  collMod: "collection",
  index: {
    name: "productId_1", // Specify the index name
    hidden: true
  }
})
```

También puede identificar el índice por su patrón de claves.

```
db.runCommand({
  collMod: "collection",
  index: {
    keyPattern: { productId: 1 }, // Specify the index key specification document
    hidden: true
  }
})
```

Mostrar un índice existente

Para volver a hacer visible un índice oculto, utilice el `collMod` comando con el valor `hidden` establecido en. `false`

```
db.runCommand({
  collMod: "collection",
  index: { name: "productId_1", hidden: false }
})
```

Como los índices se mantienen en su totalidad mientras están ocultos, el índice está inmediatamente disponible para su uso una vez que se ha ocultado.

Uso de flujos de cambios con Amazon DocumentDB

La característica de flujos de cambios de Amazon DocumentDB (compatible con MongoDB) brinda una secuencia en orden cronológico de los eventos de actualización que se producen dentro de las colecciones de su clúster. Puede leer eventos de una secuencia de cambios para implementar muchos casos de uso diferentes, incluidos los siguientes:

- Notificación de cambio
- Full-text buscar con Amazon OpenSearch Service (OpenSearch Servicio)
- Análisis con Amazon Redshift

Las aplicaciones pueden usar los flujos de cambios para suscribirse a los cambios de datos en colecciones individuales. Los eventos de flujos de cambios se ordenan a medida que se producen en el clúster y se almacenan durante 3 horas (valor predeterminado) desde el momento de registro del evento. El período de retención se puede ampliar hasta 7 días utilizando el parámetro `change_stream_log_retention_duration`. Para modificar el período de retención de la transmisión de cambios, consulteModificación de la duración de la retención del registro de flujos de cambios.

Temas

- Operaciones admitidas
- Eventos ampliados
- Facturación
- Limitaciones
- Habilitación de flujos de cambios
- Ejemplo: uso de flujos de cambios con Python
- Búsqueda completa de documentos
- Usa $change StreamSplitLargeEvent para dividir eventos grandes en fragmentos
- Reanudación de un flujo de cambios
- Reanudar un flujo de cambios con start AtOperationTime
- Reanudar un flujo de cambios con una publicación BatchResumeToken
- Transacciones en flujos de cambios
- Modificación de la duración de la retención del registro de flujos de cambios

- Uso de flujos de cambios en instancias secundarias

Operaciones admitidas

Amazon DocumentDB admite las siguientes operaciones para flujos de cambio:

- Todos los eventos de cambio admitidos en la API `db.collection.watch()`, `db.watch()` y `client.watch()` de MongoDB.
- Búsqueda completa de documentos para actualizaciones.
- Etapas de agregación: `$match` `$project``$redact`,`$addFields`,,`$replaceRoot`, y`$changeStreamSplitLargeEvent`.
- Reanudación de un flujo de cambios desde un token de reanudación
- Reanudación de un flujo de cambios a partir de una marca de tiempo mediante `startAtOperation` (aplicable a Amazon DocumentDB 4.0 y versiones posteriores)
- Notificación de eventos de DDL para `createCollection` usar el `showExpandedEvents` parámetro (aplicable a Amazon DocumentDB 8.0 a partir de la versión 8.0.2 del motor).
- Notificación de eventos de DDL para `createIndexes` usar el `showExpandedEvents` parámetro (aplicable a Amazon DocumentDB 8.0 a partir de la versión 8.0.2 del motor).

Eventos ampliados

> **Note**
>
> Los eventos ampliados están disponibles en Amazon DocumentDB 8.0 a partir de la versión 8.0.2 del motor.

Los flujos de cambios admiten las notificaciones de eventos en el lenguaje de definición de datos (DDL) y. `createCollection` `createIndexes` Para incluir eventos ampliados, abre un cursor de flujo de cambios con la `showExpandedEvents` opción.

Por ejemplo:

mongosh Method

```
let cur = db.names.watch( [ ], {
```

```
   showExpandedEvents: true
} )
cur.next()
```

Aggregation Stage

```
let cur = db.names.aggregate( [
   { $changeStream: { showExpandedEvents: true } }
] )
cur.next()
```

Facturación

La característica de secuencias de cambios de Amazon DocumentDB está desactivada de forma predeterminada y no genera ningún cargo adicional hasta que se habilita y se utiliza. El uso de secuencias de cambios en un clúster conlleva costos adicionales de lectura y escritura de IOPS y almacenamiento. Puede utilizar la operación `modifyChangeStreams` de la API con el fin de habilitar esta característica para las colecciones del clúster. Para obtener más información acerca de los precios, consulte Precios de Amazon DocumentDB.

Limitaciones

Las secuencias de cambio tienen las siguientes limitaciones en Amazon DocumentDB:

- En Amazon DocumentDB 3.6. y Amazon DocumentDB 4.0, los flujos de cambios solo se pueden abrir desde una conexión a la instancia principal de un clúster de Amazon DocumentDB. Amazon DocumentDB 3.6 y Amazon DocumentDB 4.0 no admiten la lectura de flujos de cambios en una instancia de réplica. Al invocar la operación de la API `watch()`, debe especificar una preferencia de lectura `primary` para asegurarse de que todas las lecturas se dirigen a la instancia principal (consulte la sección Ejemplo).
- En Amazon DocumentDB 5.0, los flujos de cambios se pueden abrir tanto desde la instancia principal como desde la secundaria, incluidos los clústeres globales. Puede especificar una preferencia de lectura secundaria para redirigir los flujos de cambios a instancias secundarias. Consulte Uso de flujos de cambios en instancias secundarias para ver las limitaciones y prácticas recomendadas adicionales.
- Los eventos escritos en una secuencia de cambios de una colección están disponibles hasta 7 días (el valor predeterminado es 3 horas). Los datos de flujos de cambios se eliminan después del periodo de tiempo de conservación de los registros, aunque no se hayan realizado cambios.

- Una operación de escritura de larga duración en una colección como `updateMany` o `deleteMany` puede estancar temporalmente la escritura de eventos de secuencias de cambio hasta que se completa dicha operación de escritura de larga duración.
- Amazon DocumentDB no es compatible con el registro de operaciones de MongoDB (`oplog`).
- Con Amazon DocumentDB, debe habilitar explícitamente las secuencias de cambios en una colección determinada.
- Si el tamaño total de un evento de flujo de cambios (incluidos los datos del cambio y el documento completo, si se solicita) es mayor que`16 MB`, el cliente experimentará un error de lectura en el flujo de cambios. Para evitarlo, usa el `$changeStreamSplitLargeEvent` escenario para dividir el evento en fragmentos.
- Actualmente, el controlador Ruby no es compatible al usar `db.watch()` y `client.watch()` con Amazon DocumentDB 3.6.
- El resultado del comando `updateDescription` en los flujos de cambios es diferente en Amazon DocumentDB que en MongoDB cuando el valor actualizado del campo es el mismo que el anterior:
  - Amazon DocumentDB no devuelve ningún campo en la salida `updateDescription` si el campo proporcionado está especificado en el comando `$set` y su valor objetivo ya es igual al valor de origen.
  - MongoDB devuelve el campo en la salida, incluso si el valor especificado es igual al valor actual.

Habilitación de flujos de cambios

Puede habilitar las secuencias de cambios de Amazon DocumentDB para todas las colecciones contenidas en una base de datos determinada o solamente para las colecciones seleccionadas. Los siguientes ejemplos muestran cómo habilitar las secuencias de cambios para diferentes casos de uso mediante el intérprete de comandos de mongo. Las cadenas vacías se tratan como comodines al especificar los nombres de las bases de datos y de las colecciones.

```
//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});
```

Las secuencias de cambios se habilitarán para una colección si se cumple alguna de las siguientes condiciones:

- Tanto la base de datos como la colección están habilitadas explícitamente.
- La base de datos que contiene la colección está habilitada.
- Todas las bases de datos están habilitadas.

La eliminación de una colección de una base de datos no deshabilita las secuencias de cambios para esa colección si la base de datos principal también tiene activadas las secuencias de cambios o si todas las bases de datos del clúster están habilitadas. Si se crea una nueva colección con el mismo nombre que la colección eliminada, las secuencias de cambios se habilitarán para esa colección.

Puede enumerar todas las secuencias de cambios habilitadas del clúster mediante la etapa de canalización de agregación `$listChangeStreams`. Todas las etapas de agregación admitidas por Amazon DocumentDB se pueden utilizar en la canalización para un procesamiento adicional. Si se deshabilita una colección que antes estaba habilitada, no aparecerá en la salida de `$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:{}}));
```

Ejemplo: uso de flujos de cambios con Python

El siguiente es un ejemplo del uso de una secuencia de cambios de Amazon DocumentDB con Python a nivel de la colección.

```
import os
import sys
from pymongo import MongoClient, ReadPreference

username = "DocumentDBusername"
password = <Insert your password>

clusterendpoint = "DocumentDBClusterEndpoint”
client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='global-bundle.pem')

db = client['bar']

#While ‘Primary’ is the default read preference, here we give an example of
#how to specify the required read preference when reading the change streams
coll = db.get_collection('foo', read_preference=ReadPreference.PRIMARY)
#Create a stream object
```

```
stream = coll.watch()
#Write a new document to the collection to generate a change event
coll.insert_one({'x': 1})
#Read the next change event from the stream (if any)
print(stream.try_next())

"""
Expected Output:
{'_id': {'_data': '015daf94f600000002010000000200009025'},
'clusterTime': Timestamp(1571788022, 2),
'documentKey': {'_id': ObjectId('5daf94f6ea258751778163d6')},
'fullDocument': {'_id': ObjectId('5daf94f6ea258751778163d6'), 'x': 1},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'insert'}
"""

#A subsequent attempt to read the next change event returns nothing, as there are no
 new changes
print(stream.try_next())

"""
Expected Output:
None
"""

#Generate a new change event by updating a document
result = coll.update_one({'x': 1}, {'$set': {'x': 2}})
print(stream.try_next())

"""
Expected Output:
{'_id': {'_data': '015daf99d400000001010000000100009025'},
'clusterTime': Timestamp(1571789268, 1),
'documentKey': {'_id': ObjectId('5daf9502ea258751778163d7')},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'update',
'updateDescription': {'removedFields': [], 'updatedFields': {'x': 2}}}
"""
```

El siguiente es un ejemplo del uso de una secuencia de cambios de Amazon DocumentDB con Python a nivel de la base de datos.

```
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'}
"""
```

Búsqueda completa de documentos

El evento de cambio de actualización no incluye el documento completo; incluye tan solo el cambio realizado. Si su caso de uso requiere el documento completo afectado por una actualización, puede habilitar la búsqueda completa de documentos al abrir la secuencia.

El documento `fullDocument` de un evento de secuencias de cambios de actualización representa la versión más reciente del documento actualizado en el momento de la búsqueda de documentos. Si se han producido cambios entre la operación de actualización y la búsqueda de `fullDocument`, es posible que el documento `fullDocument` no represente el estado del documento en el momento de la actualización.

Para crear un objeto de flujo con la búsqueda de actualizaciones habilitada, use este ejemplo:

```
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()
```

El resultado del objeto de flujo tendrá un aspecto similar a este:

```
{'_id': {'_data': '015daf9b7c000000010100000000100009025'},
'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}}}
```

Usa $change StreamSplitLargeEvent para dividir eventos grandes en fragmentos

> **Note**
>
> La StreamSplitLargeEvent fase $change está disponible en Amazon DocumentDB 8.0 a partir de la versión 8.0.2 del motor.

Cuando utiliza el `$changeStreamSplitLargeEvent` escenario, los eventos de cambio de más de 16 MB se dividen en fragmentos que se devuelven de forma secuencial a través del cursor del flujo de cambios. Los eventos de menos de 16 MB se devuelven sin dividir. La fase divide un evento según los límites de sus campos de nivel superior; no se divide en un solo campo ni se convierte en subdocumentos de forma recursiva. Por este motivo, si un solo campo de nivel superior supera en sí mismo los 16 MB, la etapa no puede producir un fragmento válido para él y, en su lugar, devuelve un error.

Como práctica recomendada, recomendamos evitar su uso `$changeStreamSplitLargeEvent` cuando sea posible, reduciendo el tamaño del evento.

La `$changeStreamSplitLargeEvent` etapa solo se puede usar en una `$changeStream` canalización y solo se debe usar una vez como última etapa.

Cada fragmento tiene lo suyo `resumeToken` y lo suyo`postBatchResumeToken`. Al reanudar desde un fragmento que no es definitivo (utilizando un token de currículum normal o una publicaciónBatchResumeToken), se obtiene el fragmento siguiente. En el caso de que el token corresponda al último fragmento de un evento, el cursor del flujo de cambios mostrará el evento posterior.

Reanudación de un flujo de cambios

Puede reanudar una secuencia de cambios más tarde mediante un token de reanudación, que es igual al campo `_id` del último documento de evento de cambio recuperado.

```
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}}
"""
```

Reanudar un flujo de cambios con start **`AtOperationTime`**

Puede reanudar un flujo de cambios más adelante a partir de una marca de tiempo determinada utilizando `startAtOperationTime`.

> **Note**
>
> `startAtOperationTime` se puede utilizar en Amazon DocumentDB 4.0 y versiones posteriores. Cuando se utiliza `startAtOperationTime`, el cursor del flujo de cambios solo devolverá los cambios que se hayan producido en la marca de tiempo especificada o después de ella. Los comandos `startAtOperationTime` y `resumeAfter` se excluyen mutuamente y, por lo tanto, no se pueden usar juntos.

```
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}}
"""
```

Reanudar un flujo de cambios con una publicación **`BatchResumeToken`**

El flujo de cambios de Amazon DocumentDB ahora devuelve un campo adicional denominado `postBatchResumeToken`. Este campo lo devuelven el comando `$changestream` y el comando `getMore`.

Ejemplo del comando `$changestream` en Python:

```
db.command({"aggregate": "sales", "pipeline": [{ "$changeStream": {}}], "cursor":
 {"batchSize": 1}})
```

Resultado previsto:

```
cursor" : {
   "firstBatch" : [ ],
   "postBatchResumeToken" : {"_data" : "0167c8cbe60000000004"},
   "id" : NumberLong("9660788144470"),
   "ns" : "test.sales"
}
```

Ejemplo del comando `getMore` en Python:

```
db.command({"getMore": NumberLong(<cursor id>), "collection": "sales", "batchSize":
 1 })
```

Resultado previsto

```
cursor" : {
   "nextBatch" : [ ],
   "postBatchResumeToken" : {"_data" : "0167c8cbe60000000004"},
```

```
    "id" : NumberLong("9660788144470"),
    "ns" : "test.sales"
}
```

El campo `postBatchResumeToken` se puede usar para abrir nuevos cursores del flujo de cambios en el campo `resumeAfter`, de forma similar a como se usa el token de reanudación.

Abra un flujo que comience después del `postBatchResumeToken` seleccionado:

```
post_batch_resume_token = output['cursor']['postBatchResumeToken']
stream = db.watch(full_document='updateLookup', resume_after=post_batch_resume_token)
```

A diferencia de un token de reanudación normal, que siempre corresponde a una entrada del registro de operaciones (oplog) que refleja un evento real, `postBatchResumeToken` corresponde a una entrada del oplog hasta la que el flujo de cambios ha escaneado en el servidor y que no es necesariamente un cambio coincidente.

Si intenta reanudar con un token de reanudación normal antiguo, la base de datos tendrá que escanear todas las entradas del oplog entre la marca de tiempo especificada y el momento actual. Esto puede generar muchas consultas internas y cada subconsulta se escanea durante un breve período de tiempo. Esto provocará un aumento en el uso de la CPU y degradará el rendimiento de la base de datos. Si se reanuda con el último `postBatchResumeToken`, se omite el escaneo de entradas del oplog que no coincidan.

Transacciones en flujos de cambios

Los eventos del flujo de cambios no contendrán eventos de transacciones and/or abortadas no confirmadas. Por ejemplo, si inicia una transacción con una operación de `INSERT` y una operación de `UPDATE`, y la operación de `INSERT` se realiza correctamente, pero no así la operación de `UPDATE`, la transacción se revertirá. Como esta transacción se ha revertido, tu flujo de cambios no contendrá ningún evento de esta transacción.

Modificación de la duración de la retención del registro de flujos de cambios

Puede modificar la duración de la retención del registro del flujo de cambios para que esté entre 1 hora y 7 días utilizando el Consola de administración de AWS o el. AWS CLI

Using the Consola de administración de AWS

Modificación de la duración de la retención del registro de secuencias de cambios

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Grupos de parámetros.

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En el panel Parameter groups (Grupos de parámetros), elija el grupo de parámetros de clúster asociado a su clúster. Para identificar el grupo de parámetros de clúster asociado a su clúster, consulte Determinación del grupo de parámetros de un clúster de Amazon DocumentDB.
4. La página resultante muestra los parámetros y sus detalles correspondientes para el grupo de parámetros de su clúster. Seleccione el parámetro `change_stream_log_retention_duration`.
5. En la parte superior derecha de la página, elija Edit (Editar) para cambiar el valor del parámetro. El parámetro `change_stream_log_retention_duration` se puede modificar para que abarque entre 1 hora y 7 días.
6. Realice el cambio y, a continuación, elija Modify cluster parameter (Modificar parámetro de clúster) para guardar los cambios. Para descartar los cambios, selecciona Cancel (Cancelar).

Using the AWS CLI

Para modificar los parámetros de un grupo de parámetros de clúster `change_stream_log_retention_duration`, utilice la operación `modify-db-cluster-parameter-group` con los siguientes parámetros:

- **`--db-cluster-parameter-group-name`**: obligatorio. El nombre del grupo de parámetros de clúster que va a modificar. Para identificar el grupo de parámetros de clúster asociado a su clúster, consulte Determinación del grupo de parámetros de un clúster de Amazon DocumentDB.

- **`--parameters`**: obligatorio. Los parámetros que está modificando. Cada entrada de parámetro debe incluir lo siguiente:
  - **`ParameterName`**: el nombre del grupo de parámetros de clúster que va a modificar. En este caso, es `change_stream_log_retention_duration`
  - **`ParameterValue`**: el valor nuevo de este parámetro de clúster.
  - **`ApplyMethod`**: cómo desea que se apliquen los cambios a este parámetro. Los valores permitidos son `immediate` y `pending-reboot`.

> **Note**
>
> Los parámetros con el `ApplyType` de `static` deben tener un `ApplyMethod` de `pending-reboot`.

1. Para cambiar los valores del parámetro `change_stream_log_retention_duration`, ejecute el siguiente comando y reemplace `parameter-value` por el valor que desea que tenga el parámetro.

   Para 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"
   ```

   Para 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"
   ```

   La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

   ```
   {
       "DBClusterParameterGroupName": "sample-parameter-group"
   ```

```
}
```

2. Espere al menos 5 minutos.
3. Enumere los valores de parámetros de `sample-parameter-group` para garantizar que se han realizado los cambios.

   Para Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

   Para Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

   La salida de esta operación será similar a lo que se indica a continuación (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 retención del registro de flujos de cambios no eliminará los registros anteriores al `change_stream_log_retention_duration` valor configurado hasta que el tamaño del registro sea superior a (**>**) 51.200 MB.

Uso de flujos de cambios en instancias secundarias

Para empezar a utilizar el flujo de cambios en las instancias secundarias, abra el cursor del flujo de cambios con `readPreference` como instancia secundaria.

Puede abrir un cursor de flujo de cambios para observar los cambios en una colección específica o en todas las colecciones de un clúster o base de datos. Puede abrir un cursor de flujo de cambios en cualquier instancia de Amazon DocumentDB y recuperar documentos del flujo de cambios tanto de las instancias de escritura como de lectura. Puede compartir los tokens de flujo de cambios (por ejemplo, `resumeToken` o `startOperationTime`) entre distintos cursores del flujo de cambios abiertos en una instancia de escritura y lectura.

Ejemplo

```
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)
```

Directrices y limitaciones para los flujos de cambios en las instancias secundarias

- Los eventos del flujo de cambios deben replicarse desde la instancia principal a las instancias secundarias. Puede supervisar el retraso con respecto a la `DBInstanceReplicaLag` métrica en Amazon CloudWatch.
- Es posible que las marcas de tiempo de las instancias secundarias no estén siempre sincronizadas con las de la instancia principal. En este caso, espere retrasos en la marca de tiempo de la instancia secundaria para que pueda ponerse al día. Como práctica recomendada, usa `startAtOperationTime` o `resumeToken` inicia el reloj en la instancia secundaria.
- Es posible que obtenga un rendimiento inferior en las instancias secundarias en comparación con la instancia principal si el tamaño del documento es grande; está haciendo `fullDocumentLookup` y hay una gran carga de trabajo de escritura simultánea en la instancia principal. Como práctica recomendada, monitorea la proporción de aciertos de la caché de búfer en la secundaria y asegúrate de que la proporción de aciertos de la caché de búfer sea alta.

Transacciones en Amazon DocumentDB

Amazon DocumentDB (compatible con MongoDB) admite transacciones en Amazon DocumentDB 4.0 y versiones posteriores. Puede realizar transacciones en varios documentos, estados de cuenta, colecciones y bases de datos. Las transacciones simplifican el desarrollo de aplicaciones al permitirle realizar operaciones atómicas, consistentes, aisladas y duraderas (ACID) en uno o más documentos dentro de un clúster de Amazon DocumentDB. Los casos de uso más comunes de las transacciones incluyen el procesamiento financiero, el cumplimiento y la gestión de pedidos y la creación de juegos para varios jugadores.

No hay ningún costo adicional para las transacciones. Solo paga por los dispositivos iOS de lectura y escritura que consuma como parte de las transacciones.

Temas

- Requisitos
- Prácticas recomendadas
- Limitaciones
- Supervisión y diagnóstico
- Nivel de aislamiento de transacciones
- Casos de uso
- Comandos admitidos
- Capacidades no compatibles
- Sesiones
- Errores de la transacción

Requisitos

Para utilizar la característica de transacciones, debe cumplir los siguientes requisitos:

- Debe usar Amazon DocumentDB 4.0 o una versión posterior.
- Debe usar un controlador compatible con MongoDB 4.0 o posterior.

Prácticas recomendadas

Estas son algunas de las prácticas recomendadas para que pueda aprovechar al máximo las transacciones con Amazon DocumentDB.

- Confirme o cancele siempre la transacción una vez que se haya completado. Dejar una transacción incompleta agota los recursos de la base de datos y puede provocar conflictos de escritura.
- Se recomienda limitar las transacciones al menor número de comandos necesario. Si tiene transacciones con varios estados de cuenta que se pueden dividir en varias transacciones más pequeñas, es recomendable hacerlo para reducir la probabilidad de que se agote el tiempo de espera. Procure siempre de crear transacciones cortas, no lecturas prolongadas.

Limitaciones

- Amazon DocumentDB no admite cursores en una transacción.

- Amazon DocumentDB no puede crear nuevas colecciones en una transacción y no puede crear recopilaciones no existentes query/update .
- Document-level los bloqueos de escritura están sujetos a un tiempo de espera de 1 minuto, que el usuario no puede configurar.
- En las versiones del motor anteriores a la 8.0.2, no se admiten las escrituras reintentables. A partir de la versión 8.0.2 del motor, se admiten operaciones de escritura reintentables fuera de las transacciones, en las que cada escritura individual afecta como máximo a un documento, y las operaciones de confirmación y anulación de transacciones se pueden volver a intentar. Para obtener más información, consulte Escrituras que se pueden volver a intentar en Amazon DocumentDB.
- Cada instancia de Amazon DocumentDB tiene un límite superior en cuanto al número de transacciones simultáneas que se pueden abrir en la instancia. Para obtener más información, consulte Cuotas de instancias.
- Para una transacción determinada, el tamaño del registro de transacciones debe ser inferior a 32 MB.
- Amazon DocumentDB admite `count()` dentro de transacciones, pero no todos los controladores admiten esta capacidad. Una alternativa es utilizar la API `countDocuments()`, que convierte la consulta de recuento en una consulta de agregación en el lado del cliente.
- Las transacciones tienen un límite de ejecución de un minuto y las sesiones tienen un tiempo de espera de 30 minutos. Si se agota el tiempo de espera de una transacción, se cancelará y cualquier comando posterior que se ejecute dentro de la sesión para la transacción existente generará el siguiente error:

```
WriteCommandError({
"ok" : 0,
"operationTime" : Timestamp(1603491424, 627726),
"code" : 251,
"errmsg" : "Given transaction number 0 does not match any in-progress transactions."
}
```

Supervisión y diagnóstico

Gracias a la compatibilidad con las transacciones de Amazon DocumentDB 4.0, se agregaron CloudWatch métricas adicionales para ayudarlo a monitorear sus transacciones.

Métricas nuevas CloudWatch

- `DatabaseTransactions`: el número de transacciones abiertas realizadas en un período de un minuto.
- `DatabaseTransactionsAborted`: el número de transacciones canceladas realizadas en un período de un minuto.
- `DatabaseTransactionsMax`: el número máximo de transacciones abiertas realizadas en un período de un minuto.
- `TransactionsAborted`: el número de transacciones canceladas en una instancia en un período de un minuto.
- `TransactionsCommitted`: el número de transacciones confirmadas en una instancia en un período de un minuto.
- `TransactionsOpen`: el número de transacciones abiertas en una instancia en un período de un minuto.
- `TransactionsOpenMax`: el número máximo de transacciones abiertas en una instancia en un período de un minuto.
- `TransactionsStarted`: el número de transacciones canceladas en una instancia en un período de un minuto.

> **Note**
>
> Para obtener más CloudWatch métricas de Amazon DocumentDB, vaya aSupervisión de Amazon DocumentDB con CloudWatch.

Además, se agregaron nuevos campos a `currentOp` `lsid`, `transactionThreadId`, y un nuevo estado para "`idle transaction`" y las transacciones `serverStatus`: `currentActive`, `currentInactive`, `currentOpen`, `totalAborted`, `totalCommitted` y `totalStarted`.

Nivel de aislamiento de transacciones

Al iniciar una transacción, puede especificar tanto la `readConcern` `writeConcern` como se muestra en el siguiente ejemplo:

```
mySession.startTransaction({readConcern: {level: 'snapshot'}, writeConcern:
{w: 'majority'}});
```

Para `readConcern`, Amazon DocumentDB admite el aislamiento de instantáneas de forma predeterminada. Si se especifica una `readConcern` local, disponible o mayoritario, Amazon DocumentDB actualizará el nivel de `readConcern` a instantánea. Amazon DocumentDB no admite el `readConcern` linealizable y, si se especifica una preocupación de lectura de este tipo, se producirá un error.

Para `writeConcern`, Amazon DocumentDB admite la mayoría de forma predeterminada y se logra un quórum de escritura cuando se conservan cuatro copias de los datos en tres zonas de disponibilidad. Si se especifica una `writeConcern` inferior, Amazon DocumentDB la actualizará la `writeConcern` a la mayoría. Además, todas las escrituras de Amazon DocumentDB se registran en un diario y no se puede deshabilitar.

Casos de uso

En esta sección, analizaremos dos casos de uso de las transacciones: varias declaraciones y varias recopilaciones.

Multi-Statement Transacciones

Las transacciones de Amazon DocumentDB son de varios estados, lo que significa que puede escribir una transacción que abarque varios estados con una confirmación o reversión explícita. Puede agrupar acciones de `insert`, `update`, `delete` y `findAndModify` como una sola operación atómica.

Un caso de uso común para las transacciones con varios estados de cuenta es una transacción de débito-crédito. Por ejemplo: le debe dinero a un amigo de unas prendas de ropa. Por lo tanto, debe cargar (retirar) $500 de su cuenta y acreditar $500 (depósito) en la cuenta de su amigo. Para realizar esa operación, debe realizar las operaciones de débito y crédito en una sola transacción para garantizar la atomicidad. Si lo hace, evitará que se debiten $500 de su cuenta, pero no se acrediten en la cuenta de su amigo. Este es el aspecto que tendría este caso de 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 transacciones

Nuestras transacciones también son de cobro múltiple, lo que significa que se pueden utilizar para realizar múltiples operaciones dentro de una sola transacción y en varios cobros. Esto proporciona una visión coherente de los datos y mantiene la integridad de los mismos. Cuando se ejecutan los comandos como un solo <>, las transacciones son ejecuciones de todo o nada, es decir, todas se ejecutarán correctamente o todas fallarán.

A continuación, se muestra un ejemplo de transacciones de varios cobros, en las que se utiliza el mismo escenario y los mismos datos del ejemplo para las transacciones con varios estados de cuenta.

```
// *** 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
```

Ejemplos de API de transacciones para la API de devolución de llamada

La API de devolución de llamada solo está disponible para controladores 4,2 o más.

Javascript

El siguiente código muestra cómo utilizar la API de transacciones de 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

El siguiente código muestra cómo utilizar la API de transacciones de 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#

El siguiente código muestra cómo utilizar la API de transacciones de 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

El siguiente código muestra cómo utilizar la API de transacciones de 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

El siguiente código muestra cómo utilizar la API de transacciones de 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

El siguiente código muestra cómo utilizar la API de transacciones de 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

El siguiente código muestra cómo utilizar la API de transacciones de 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

El siguiente código muestra cómo utilizar la API de transacciones de 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
```

Ejemplos de API de transacciones para la API de Core

Javascript

El siguiente código muestra cómo utilizar la API de transacciones de 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#

El siguiente código muestra cómo utilizar la API de transacciones de 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

El siguiente código muestra cómo utilizar la API de transacciones de 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

El siguiente código muestra cómo utilizar la API de transacciones de 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

El siguiente código muestra cómo utilizar la API de transacciones de 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

El siguiente código muestra cómo utilizar la API de transacciones de 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

El siguiente código muestra cómo utilizar la API de transacciones de 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

El siguiente código muestra cómo utilizar la API de transacciones de 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
```

Comandos admitidos

Comando	compatible
`abortTransaction`	Sí
`commitTransaction`	Sí
`endSessions`	Sí
`killSession`	Sí
`killAllSession`	Sí
`killAllSessionsByPattern`	No
`refreshSessions`	No
`startSession`	Sí

Capacidades no compatibles

Métodos	Etapas o comandos
`db.collection.aggregate()`	`$collStats` `$currentOp`

Métodos	Etapas o comandos

	`$indexStats`
	
	`$listSessions`
	
	`$out`

`db.collection.count()`
 	
`db.collection.countDocuments()`
	`$where`
	
	`$near`
	
	`$nearSphere`

`db.collection.insert()`	`insert` no se admite si no se ejecuta en una colección existente. Este método es compatible si se dirige a una colección preexistente.

Sesiones

Las sesiones de MongoDB son un marco que se utiliza para admitir escrituras reintentables, coherencia causal, transacciones y administrar operaciones en bases de datos. Cuando se crea una sesión, el cliente genera un identificador de sesión lógico (lsid) que se utiliza para etiquetar todas las operaciones de esa sesión al enviar comandos al servidor.

Amazon DocumentDB admite el uso de sesiones para habilitar las transacciones, pero no admite la coherencia causal. A partir de la versión 8.0.2 del motor, Amazon DocumentDB admite reintentos de escritura fuera de las transacciones para operaciones de escritura en las que cada escritura individual afecta como máximo a un documento. En las versiones anteriores del motor, no se admiten escrituras que se puedan volver a intentar. Para obtener más información, consulte Escrituras que se pueden volver a intentar en Amazon DocumentDB.

Al utilizar transacciones en Amazon DocumentDB, una transacción se iniciará desde una sesión mediante la API `session.startTransaction()` y una sesión admite una sola transacción a la vez. Del mismo modo, las transacciones se completan mediante las API commit (`session.commitTransaction()`) o abort (`session.abortTransaction()`).

Coherencia causal

La coherencia causal garantiza que, en una sola sesión de cliente, el cliente observará una coherencia de lectura tras escritura, es monoatómica reads/writes, y las escrituras seguirán a las lecturas. Estas garantías se aplican a todas las instancias de un clúster, no solo a la principal. Amazon DocumentDB no admite la coherencia causal y la siguiente afirmación generará un error.

```
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()
```

Puede deshabilitar la coherencia causal dentro de una sesión. Tenga en cuenta que, si lo hace, podrá utilizar el marco de la sesión, pero no garantizará la coherencia causal de las lecturas. Al usar Amazon DocumentDB, las lecturas de la instancia principal serán consistentes de lectura tras escritura y las lecturas de las instancias de réplica serán consistentes eventualmente. Las transacciones son el principal caso de uso para utilizar las sesiones.

```
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 }
```

Reintento de las escrituras

Con las escrituras que se pueden volver a intentar, el controlador reintenta automáticamente una operación de escritura una vez si se produce un error de red o si el controlador no puede encontrar la principal. A partir de la versión 8.0.2 del motor, Amazon DocumentDB admite escrituras que se pueden volver a intentar. En las versiones anteriores del motor, las escrituras que se pueden volver a intentar se deben deshabilitar configurándolas en la cadena de conexión. `retryWrites=false` Para obtener más información, consulte Escrituras que se pueden volver a intentar en Amazon DocumentDB.

Errores de la transacción

Cuando se utilizan transacciones, hay situaciones en las que se puede devolver un error que indique que el número de transacción no coincide con ninguna transacción en curso.

El error se puede generar en al menos dos escenarios diferentes:

- Una vez transcurrido el tiempo de espera de la transacción de un minuto.
- Tras el reinicio de la instancia (debido a una aplicación de un parche, una recuperación tras un error, etc.), es posible que se produzca este error incluso en los casos en que la transacción se haya hecho correctamente. Durante el reinicio de una instancia, la base de datos no puede diferenciar entre una transacción que se completó correctamente y una transacción que se canceló. En otras palabras, el estado de finalización de la transacción es ambiguo.

La mejor forma de solucionar este error es hacer que las actualizaciones transaccionales sean idempotentes, por ejemplo, utilizando el `$set` mutador en lugar de una operación. increment/ decrement Consulte a continuación:

```
{ "ok" : 0,
"operationTime" : Timestamp(1603938167, 1),
"code" : 251,
"errmsg" : "Given transaction number 1 does not match any in-progress transactions."
}
```

Escrituras que se pueden volver a intentar en Amazon DocumentDB

A partir de la versión 8.0.2 del motor, Amazon DocumentDB admite escrituras que se pueden volver a intentar. Cuando una operación de escritura falla debido a un error transitorio de red o a una elección primaria, el controlador puede volver a intentar la operación automáticamente una sola vez. Amazon DocumentDB deduplica la escritura reintentada para que la operación se aplique como máximo una vez, preservando la idempotencia.

Las escrituras reintentables requieren un MongoDB-compatible controlador que sea compatible con el protocolo de escritura reintentable. La mayoría de MongoDB-compatible los controladores actuales permiten realizar escrituras reintentables de forma predeterminada (`retryWrites=true`en la cadena de conexión). Si está actualizando el motor 8.0.2 desde una versión anterior, puede eliminarla `retryWrites=false` de la cadena de conexión para habilitar esta función.

Requisitos

Para utilizar escrituras que se puedan volver a intentar, debe cumplir los siguientes requisitos:

- Versión 8.0.2 o posterior del motor Amazon DocumentDB.
- Un MongoDB-compatible controlador que admite escrituras que se pueden volver a intentar. Para obtener la versión mínima, consulte la documentación de su controlador.
- La cadena de conexión debe incluir`retryWrites=true`, o el controlador debe tener la opción predeterminada de reintentar las escrituras (lo hacen la mayoría de los controladores actuales).

Operaciones admitidas

Las escrituras reintentables se aplican a las operaciones de escritura en las que cada escritura individual afecta como máximo a un documento. Las operaciones por lotes cumplen los requisitos, ya que cada escritura del lote se deduplica por separado.

Se pueden volver a intentar las siguientes operaciones de escritura:

- `insertOne`
- `insertMany`
- `updateOne`
- `deleteOne`

- `findOneAndUpdate`
- `findOneAndDelete`
- `findOneAndReplace`
- `bulkWrite`(cuando se compone de`insertOne`, `updateOne`de`deleteOne`, u `replaceOne` operaciones)

En el `insertMany` caso de y`bulkWrite`, cada documento del lote se deduplica de forma individual. Si es necesario volver a intentarlo, solo se insertan los documentos que aún no se hayan aplicado.

> **Note**
>
> `commitTransaction`y también `abortTransaction` se pueden volver a intentar, como comandos de control de transacciones y no como escrituras. Las escrituras individuales dentro de una transacción no se pueden volver a intentar. Para obtener más información, consulte Limitaciones.

Limitaciones

Las siguientes limitaciones se aplican a las escrituras que se pueden volver a intentar en Amazon DocumentDB:

- `updateMany`y no se pueden volver a `deleteMany` intentar.
- Las escrituras en transacciones con varios estados de cuenta no se pueden volver a intentar. Las operaciones de confirmación y cancelación de transacciones se pueden volver a intentar por separado.
- Los documentos deben incluir un `_id` campo para las inserciones que se puedan volver a intentar.
- Las escrituras que se pueden volver a intentar solo están disponibles en la versión 8.0.2 y posteriores del motor. En las versiones anteriores del motor, establézcala `retryWrites=false` en la cadena de conexión para evitar errores.
- Las operaciones de escritura no se pueden volver a intentar en los clústeres que utilizan la versión 1.0 del planificador de consultas.
- `findOneAndUpdate`En `findOneAndDelete` el caso de `findOneAndReplace` las operaciones que devuelven documentos de gran tamaño, las escrituras que se pueden volver a intentar pueden aumentar la latencia de escritura, ya que todo el documento resultante se almacena en caché para

su deduplicación. Si el rendimiento de escritura es fundamental y los documentos devueltos son grandes, considere la posibilidad de configurar `retryWrites=false` esas cargas de trabajo.

Habilitar las escrituras que se pueden volver a intentar

Para habilitar las escrituras que se pueden volver a intentar, incluye `retryWrites=true` en la cadena de conexión:

```
mongodb://<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=true
```

Si el controlador tiene el valor predeterminado`retryWrites=true`, puedes eliminar cualquier contenido explícito `retryWrites=false` de tu cadena de conexión.

Para ver el tutorial completo sobre la cadena de conexión, consulte. Conexión mediante programación a Amazon DocumentDB

Cómo funcionan las escrituras que se pueden volver a intentar

Cuando la aplicación envía una escritura con`retryWrites=true`, el controlador adjunta un identificador de sesión lógico (`lsid`) y un número de transacción (`txnNumber`) al comando de escritura. Amazon DocumentDB usa estos identificadores para deduplicar las escrituras:

- En el primer intento, Amazon DocumentDB ejecuta la escritura y almacena en caché el resultado.
- Si el controlador vuelve a intentar la misma escritura (la misma `lsid` `txnNumber`), Amazon DocumentDB devuelve el resultado almacenado en caché sin volver a ejecutar la escritura.
- Amazon DocumentDB acepta los reintentos durante 60 minutos después de la escritura original. Los controladores vuelven a intentarlo inmediatamente, por lo que este período es mucho más largo de lo que necesita un controlador.

Si se produce un reintento más de 60 minutos después de la escritura original, Amazon DocumentDB ya no tiene el resultado almacenado en caché y ejecuta la escritura como una operación nueva.

Gestión de errores

Los siguientes errores son específicos de las escrituras que se pueden volver a intentar en Amazon DocumentDB:

Código de error	Name	Description (Descripción)
225	TransactionTooOld	Ya se ha confirmado un número de transacción más nuevo para la misma sesión y declaración, por lo que este reintento es demasiado antiguo para aplicarlo.
301	No se admiten las escrituras que se pueden volver a intentar	La escritura incluía campos de escritura reintenta bles, pero la versión del motor no admite escritura s reintentables o la operación no se puede volver a intentar. Se establece `retryWrites=false` cuando se conecta a versiones del motor anteriores a la 8.0.2.

Migración desde RetryWrites=false

Si estás actualizando a la versión 8.0.2 del motor y actualmente la utilizas en tus cadenas de conexión: `retryWrites=false`

1. Actualiza tu clúster a la versión 8.0.2 del motor o a una posterior.
2. Elimine `retryWrites=false` las cadenas de conexión de su aplicación o cámbielas por. `retryWrites=true`
3. No es necesario cambiar el código de la aplicación. El controlador gestiona los reintentos automáticamente.

Uso de la intercalación en Amazon DocumentDB 8.0

Amazon DocumentDB 8.0 ahora admite la intercalación. La intercalación le permite configurar reglas específicas del idioma para la comparación de cadenas. Con la intercalación, puede especificar reglas para las comparaciones que distingan mayúsculas de minúsculas o especificar una configuración regional de idioma. La intercalación se puede configurar a nivel de colección o de índice en DocumentDB 8.0. Cuando se utiliza la intercalación en DocumentDB, internamente se crea un documento de intercalación con los siguientes parámetros.

```
{
locale: string,
```

```
caseLevel: boolean,
caseFirst: string,
strength: int,
numericOrdering: boolean,
alternate: string,
maxVariable: string,
backwards: boolean,
normalization: boolean
}
```

Limitaciones

La intercalación tiene las siguientes limitaciones en Amazon DocumentDB:

- La intercalación es compatible con el planificador v3 disponible en Amazon DocumentDB 8.0. El cambio a plannerv2 o plannerv1 puede provocar comportamientos incoherentes, incluidos errores de tipo «índice no encontrado».
- Debido a las diferencias inherentes a las bibliotecas, si se exporta una colección de mongodb con intercalación, tendrá que actualizar los archivos.bson de metadatos y cambiar su versión de 57.1 a 60.2 antes de realizar la migración.
- En raras ocasiones, la configuración de intercalación puede superar el límite interno de número de caracteres y provocar el siguiente error. «Error: el documento de intercalación tiene más atributos no predeterminados de los admitidos. Reduzca el número de opciones». En este caso, intente reducir las opciones que proporciona en el documento de cotejo o, si lo prefiere, puede intentar utilizar los valores predeterminados siempre que sea posible.

Uso de vistas en Amazon DocumentDB 8.0

- Amazon DocumentDB 8.0 ahora admite vistas. Las vistas funcionan como colecciones virtuales que presentan datos en función de operaciones de agregación especificadas. Al crear una vista, se define una consulta que transforma los datos de una o más colecciones de fuentes. Amazon DocumentDB 8.0 ejecuta esta consulta cada vez que se accede a la vista, sin consumir recursos de almacenamiento adicionales. A diferencia de las colecciones estándar, las vistas de Amazon DocumentDB 8.0 no almacenan los documentos en disco, lo que las convierte en una solución eficaz para presentar datos transformados o filtrados a las aplicaciones. Para crear una vista en Amazon DocumentDB, puede utilizar el comando CreateView o el método auxiliar db.createView ():

```
db.createView("viewName","sourceCollection",
[
    { $match: { status: "active" } },
    { $project: { _id: 1, name: 1, email: 1 } }
]
)
```

Esto crea una vista denominada «viewName» basada en «SourceCollection» que solo incluye los documentos activos y proyecta únicamente los campos _id, name y correo electrónico. Las vistas de Amazon DocumentDB son de solo lectura. Las operaciones de escritura en las vistas devolverán un error. Para obtener un rendimiento óptimo con conjuntos de datos de gran tamaño, puede estructurar sus canalizaciones de visualización para maximizar la eficiencia. Para los procesos de agregación complejos, se recomienda utilizar la etapa $match como primera etapa o al principio del proceso para reducir la cantidad de documentos que deben procesarse las etapas posteriores y, de este modo, mejorar el rendimiento de las consultas.

Prácticas recomendadas

A continuación se detallan algunas de las prácticas recomendadas que se deben seguir con Views.

- Filtrar anticipadamente: utilice las etapas de $match al principio de las canalizaciones de visualización para reducir la cantidad de datos procesados.
- Evite las agregaciones complejas: para las vistas con agregaciones complejas a las que se accede con frecuencia, considere la posibilidad de crear una colección independiente con resultados precalculados que se actualicen periódicamente.
- Planificación de índices: asegúrese de que los campos utilizados en las canalizaciones de vistas, especialmente en las operaciones $match y $sort, estén indexados correctamente en la colección de origen.
- Optimización de consultas: utilice el comando explain para comprender cómo se ejecutan las consultas de visualización y optimizarlas en consecuencia.
- Alternativas para las vistas: dadas las diferencias funcionales entre las vistas de Amazon DocumentDB y MongoDB, considere la posibilidad de utilizar colecciones regulares con actualizaciones programadas como alternativa a las vistas cuando se encuentre con limitaciones.

Compatibilidad con los operadores de agregadores

Amazon DocumentDB admite muchos operadores de agregación en las definiciones de vistas y, al mismo tiempo, sigue ampliando la compatibilidad. Cuando utilice vistas, concéntrese en los siguientes operadores compatibles:

- $match para filtrar documentos
- $project para la selección y transformación de campos
- $addFields para añadir campos calculados
- $sort para ordenar los resultados
- $limit y $skip para paginación

Algunos operadores especializados, como $currentOp, $replaceRoot y $geoNear, actualmente funcionan en consultas de agregación directa en lugar de ver definiciones.

Aprovechando los índices y las vistas

Las vistas de Amazon DocumentDB 8.0 utilizan los índices de la colección subyacente. Como resultado, no puede crear, eliminar ni reconstruir índices directamente en una vista. Sin embargo, unos índices bien diseñados en la colección fuente pueden mejorar considerablemente el rendimiento de las consultas de visualización. A continuación, se indican algunos pasos para optimizar el rendimiento de las consultas en las vistas:

- Asegúrese de que existan los índices adecuados en los campos de la colección fuente utilizados en la canalización de la vista, especialmente en las operaciones $match y $sort
- Utilice el método explain () para analizar los planes de ejecución de las consultas y verificar el uso de los índices. E.g., `db.viewName.find({...}).explain()`

Inteligencia artificial generativa de Amazon DocumentDB

Amazon DocumentDB ofrece capacidades para permitir que los modelos de machine learning (ML) e inteligencia artificial (IA) generativa funcionen con los datos almacenados en Amazon DocumentDB en tiempo real. Los clientes ya no tienen que perder tiempo administrando una infraestructura independiente, escribiendo código para conectarse con otro servicio y duplicando datos de su base de datos principal.

Para obtener más información sobre la inteligencia artificial y cómo AWS puede satisfacer sus necesidades de inteligencia artificial, consulte este artículo What-is "».

Temas

- No-code aprendizaje automático con Amazon SageMaker AI Canvas
- Búsqueda vectorial para Amazon DocumentDB

No-code aprendizaje automático con Amazon SageMaker AI Canvas

Amazon SageMaker AI Canvas le permite crear sus propios AI/ML modelos sin tener que escribir una sola línea de código. Puede crear modelos de machine learning para casos de uso comunes, como la regresión y la previsión, y puede acceder a los modelos fundacionales (FM) de Amazon Bedrock y evaluarlos. También puede acceder a los FM públicos de Amazon SageMaker AI JumpStart para generar contenido, extraer y resumir textos, a fin de respaldar las soluciones de IA generativas.

Cómo crear modelos de aprendizaje automático sin código con AI Canvas SageMaker

Amazon DocumentDB ahora se integra con Amazon SageMaker AI Canvas para permitir el aprendizaje automático (ML) sin código con los datos almacenados en Amazon DocumentDB. Ahora puede crear modelos de machine learning para las necesidades de regresión y previsión y utilizar modelos fundacionales para resumir y generar contenido con datos almacenados en Amazon DocumentDB sin necesidad de escribir una sola línea de código.

SageMaker AI Canvas proporciona una interfaz visual que permite a los clientes de Amazon DocumentDB generar predicciones sin necesidad de AI/ML conocimientos especializados ni escribir una sola línea de código. Los clientes ahora pueden lanzar el espacio de trabajo de SageMaker AI Canvas desde los datos de Amazon DocumentDB Consola de administración de AWS, importarlos y unirlos para la preparación de datos y el entrenamiento de modelos. Los datos de Amazon DocumentDB ahora se pueden usar en SageMaker AI Canvas para crear y ampliar modelos que permitan predecir la pérdida de clientes, detectar fraudes, predecir fallos de mantenimiento, pronosticar métricas empresariales y generar contenido. Los clientes ahora pueden publicar y compartir ML-driven información entre equipos mediante la integración nativa de SageMaker AI Canvas con Quick. Las canalizaciones de ingestión de datos de SageMaker AI Canvas se ejecutan en instancias secundarias de Amazon DocumentDB de forma predeterminada, lo que garantiza que el rendimiento de las cargas de trabajo de ingestión de aplicaciones y de SageMaker AI Canvas no se vea obstaculizado.

Los clientes de Amazon DocumentDB pueden empezar a usar SageMaker AI Canvas accediendo a la nueva página de la consola No-Code ML de Amazon DocumentDB y conectándose a los espacios de trabajo de AI Canvas nuevos o disponibles. SageMaker

Configurar el dominio de SageMaker IA y el perfil de usuario

Puede conectarse a los clústeres de Amazon DocumentDB desde dominios de SageMaker IA que se ejecutan en modo solo de VPC. Al lanzar un dominio de SageMaker IA en su VPC, puede controlar el flujo de datos desde sus entornos de SageMaker AI Studio y Canvas. Esto te permite restringir el acceso a Internet, supervisar e inspeccionar el tráfico mediante funciones de AWS red y seguridad estándar, y conectarte a otros AWS recursos a través de los puntos de conexión de la VPC. Consulte la Guía para desarrolladores de Amazon SageMaker AI Canvas para comenzar y configurar Amazon SageMaker AI Canvas en una VPC sin acceso a Internet, que se encuentra en la Guía para desarrolladores de Amazon SageMaker AI para crear un dominio de SageMaker IA y conectarlo a su clúster de Amazon DocumentDB.

Configuración de los permisos de acceso de IAM para Amazon DocumentDB y AI Canvas SageMaker

Un usuario de Amazon DocumentDB que haya asociado `AmazonDocDBConsoleFullAccess` a su rol e identidad asociados puede acceder a la Consola de administración de AWS. Agregue las siguientes acciones al rol o la identidad antes mencionados para proporcionar acceso al aprendizaje automático sin código con Amazon SageMaker AI Canvas.

```
"sagemaker:CreatePresignedDomainUrl",
"sagemaker:DescribeDomain",
"sagemaker:ListDomains",
"sagemaker:ListUserProfiles"
```

Creación de usuarios y roles de bases de datos para SageMaker AI Canvas

Puede restringir el acceso a las acciones que los usuarios pueden hacer en bases de datos mediante el control de acceso basado en roles (RBAC) en Amazon DocumentDB. RBAC funciona otorgando uno o más roles a un usuario. Estos roles determinan las operaciones que un usuario puede realizar en los recursos de la base de datos.

Como usuario de Canvas, se conecta a una base de datos de Amazon DocumentDB con credenciales de nombre de usuario y contraseña. Puede crear una base de datos user/role para

un usuario de Canvas que tenga acceso de lectura a las bases de datos específicas mediante la funcionalidad RBAC de Amazon DocumentDB.

Por ejemplo, utilice la operación `createUser`:

```
db.createUser({
user: "canvas_user",
pwd: "<insert-password>",
roles: [{role: "read", db: "sample-database-1"}]
})
```

Esto crea un `canvas_user` que tiene permisos de lectura en la base de datos `sample-database-1`. Los analistas de Canvas pueden utilizar esta credencial para acceder a los datos del clúster de Amazon DocumentDB. Consulte Acceso a bases de datos mediante Role-Based Access Control para obtener más información.

Regiones disponibles

La integración sin código está disponible en las regiones en las que se admiten Amazon DocumentDB y Amazon SageMaker AI Canvas. Las regiones incluyen:

- us-east-1 (Norte de Virginia)
- us-east-2 (Ohio)
- us-west-2 (Oregón)
- ap-northeast-1 (Tokio)
- ap-northeast-2 (Seúl)
- ap-south-1 (Mumbai)
- ap-southeast-1 (Singapur)
- ap-southeast-2 (Sídney)
- eu-central-1 (Fráncfort)
- eu-west-1 (Irlanda)

Consulte Amazon SageMaker AI Canvas en la Guía para desarrolladores de Amazon SageMaker AI para conocer la disponibilidad más reciente en la región.

Búsqueda vectorial para Amazon DocumentDB

La búsqueda vectorial es un método utilizado en el machine learning para encontrar puntos de datos similares a un punto de datos determinado mediante la comparación de sus representaciones vectoriales mediante métricas de distancia o similitud. Cuanto más cerca estén los dos vectores del espacio vectorial, más parecidos se considerarán los elementos subyacentes. Esta técnica ayuda a captar el significado semántico de los datos. Este enfoque es útil en diversas aplicaciones, como los sistemas de recomendación, el procesamiento del lenguaje natural y el reconocimiento de imágenes.

La búsqueda vectorial para Amazon DocumentDB combina la flexibilidad y la rica capacidad de consulta de una base de datos de JSON-based documentos con la potencia de la búsqueda vectorial. Si desea utilizar sus datos actuales de Amazon DocumentDB o una estructura de datos de documentos flexible para crear casos de uso de machine learning e IA generativa, como la experiencia de búsqueda semántica, recomendación de productos, personalización, chatbots, detección de fraudes y detección de anomalías, la búsqueda vectorial para Amazon DocumentDB es la opción ideal para usted. La búsqueda vectorial está disponible en los clústeres basados en instancias de Amazon DocumentDB 5.0.

Temas

- Inserción de vectores
- Creación de un índice vectorial
- Obtención de una definición de índice
- Consulta de vectores
- Características y limitaciones
- Prácticas recomendadas

Inserción de vectores

Para insertar vectores en la base de datos de Amazon DocumentDB, puede utilizar los métodos de inserción existentes:

Ejemplo

En el siguiente ejemplo, se crea una colección de cinco documentos dentro de una base de datos de prueba. Cada documento incluye dos campos: el nombre del producto y su incrustación vectorial correspondiente.

```
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]}
]);
```

Creación de un índice vectorial

Amazon DocumentDB admite los métodos de indexación Hierarchical Navigable Small World (HNSW) e Inverted File with Flat Compression (IVFFlat). Un índice IVFFlat segrega los vectores en listas y, posteriormente, busca en un subconjunto seleccionado de esas listas que esté más cerca del vector de consulta. Por otro lado, un índice HNSW organiza los datos vectoriales en un gráfico de varias capas. Si bien HNSW tiene tiempos de creación más lentos en comparación con IVFFlat, ofrece un mejor rendimiento y recuperación de las consultas. A diferencia de IVFFlat, HNSW no implica ningún paso de entrenamiento, lo que permite generar el índice sin ninguna carga inicial de datos. En la mayoría de los casos de uso, utilice el tipo de índice HNSW para la búsqueda vectorial.

Si no crea un índice vectorial, Amazon DocumentDB hacer una búsqueda exacta del vecino más cercano, lo que garantiza una recuperación perfecta. Sin embargo, en los escenarios de producción, la velocidad es crucial. Utilice índices vectoriales, que podrían cambiar un poco de recuperación por una mayor velocidad. Es importante tener en cuenta que agregar un índice vectorial puede generar resultados de consulta diferentes.

Plantillas

Puede usar las siguientes plantillas de `createIndex` o `runCommand` para crear un índice vectorial en un campo vectorial:

Using createIndex

En algunos controladores, como mongosh y Java, el uso de los `vectorOptions` parámetros in `createIndex` puede provocar un error. En estos casos, utilice: `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

En algunos controladores, como mongosh y Java, el uso de los `vectorOptions` parámetros in `createIndex` puede provocar un error. En estos casos, utilice: `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"
      }]
  }
);
```

Parámetro	Requisito	Tipo de datos:	Description (Descripción)	Valores
`name`	opcional	cadena	Especifica el nombre del índice.	Alfanumérico

Parámetro	Requisito	Tipo de datos:	Description (Descripción)	Valores
`type`	opcional		Especifica el tipo de índice.	Compatible: hnsw o ivfflat Valor predeterminado: HNSW (con el parche del motor a partir de la versión 3.0.4574)
`dimensions`	obligatorio	entero	Especifica el número de dimensiones de los datos vectoriales.	Máximo de 2000 dimensiones.
`similarity`	obligatorio	cadena	Especifica la métrica de distancia utilizada para el cálculo de similitud.	• **`euclidean`** • **`cosine`** • **`dotProduct`**

Parámetro	Requisito	Tipo de datos:	Description (Descripción)	Valores
`lists`	obligatorio para IVFFlat	entero	Especifica el número de clústeres que el índice de IVFFlat utiliza para agrupar los datos vectoriales. La configuración recomendada es el número de documents /1000 para un máximo de 1 millón de documentos y `sqrt(# of documents )` para más de 1 millón de documentos.	Mínimo: 1 Máximo: consulte la tabla de listas por tipo de instancia que aparece en Características y limitaciones a continuación.
`m`	opcional	entero	Especifica el número máximo de conexiones para un índice HNSW	Predeterminado: 16 Rango: [2, 100]

Parámetro	Requisito	Tipo de datos:	Description (Descripción)	Valores
efConstruction	opcional	entero	Especifica el tamaño de la lista dinámica de candidatos para construir el gráfico del índice de HNSW. `efConstruction` debe ser mayor o igual que (2 * m)	Predeterminado: 64 Rango [4, 1000]

Es importante que definas el valor de los subparámetros (por ejemplo, `lists` para IVFflat `m` y `efConstruction` para HNSW) de forma adecuada accuracy/recall, ya que esto afectará al tiempo de creación y al rendimiento de la búsqueda. Un valor de lista más alto aumenta la velocidad de la consulta, ya que reduce el número de vectores de cada lista, lo que se traduce en regiones más pequeñas. Sin embargo, un tamaño de región más pequeño puede provocar más errores de recuperación, lo que se traduce en una menor precisión. En el caso de HNSW, aumentar el valor `m` y `efConstruction` aumenta la precisión, pero también se incrementa el tiempo y el tamaño de creación del índice. Consulte los siguientes ejemplos:

Ejemplos

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
    }
  }
)
```

Obtención de una definición de índice

Puede ver los detalles de los índices, incluidos los índices vectoriales, mediante el comando `getIndexes`:

Ejemplo

```
db.collection.getIndexes()
```

Ejemplo de salida

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

Consulta de vectores

Amazon DocumentDB admite dos operadores de búsqueda vectorial para consultar vectores:

Operador de búsqueda vectorial clásico

Utilice la siguiente plantilla para consultar un vector:

```
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]
      }
    }
  }
]);
```

Parámetro	Requisito	Tipo	Description (Descripción)	Valores
vectorSearch	obligatorio	operador	Se utiliza dentro del comando $search para	

Parámetro	Requisito	Tipo	Description (Descripción)	Valores
			consultar los vectores.	
vector	obligatorio	array	Indica el vector de consulta que se utilizará para buscar vectores similares.	
path	obligatorio	cadena	Define el nombre del campo vectorial.	
k	obligatorio	entero	Especifica el número de resultados que devuelve la búsqueda.	
similarity	obligatorio	cadena	Especifica la métrica de distancia utilizada para el cálculo de similitud.	• **euclidean** • **cosine** • **dotProduct**

Parámetro	Requisito	Tipo	Description (Descripción)	Valores
`probes`	opcional	entero	El número de clústeres que desea que inspeccione la búsqueda vectorial. Un valor más alto proporciona una mejor recuperación a costa de la velocidad. Se puede establecer en función del número de listas para la búsqueda exacta del vecino más cercano (momento en el que el planificador no utilizará el índice). La configuración recomendada para iniciar el refinamiento es `sqrt(# of lists)`.	Valor predeterminado: 1

Parámetro	Requisito	Tipo	Description (Descripción)	Valores
`efSearch`	opcional	entero	Especifica el tamaño de la lista dinámica de candidatos que el índice de HNSW utiliza durante la búsqueda. Un valor más alto de `efSearch` proporciona una mejor recuperación a costa de la velocidad.	Predeterminado: 40 Rango: [1, 1000]

Es importante refinar el valor de `efSearch` (HNSW) o `probes` (IVFlat) para lograr el rendimiento y la precisión deseados. Vea los siguientes ejemplos de operaciones:

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
      }
    }
  }
]);
```

Ejemplo de salida

La salida de esta operación será similar a lo que se indica a continuación:

```
{ "_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 ] }
```

Operador $VectorSearch (disponible en Amazon DocumentDB 8.0 y versiones posteriores)

Utilice la siguiente plantilla para consultar un vector:

```
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],
  }
}])
```

Características y limitaciones

Compatibilidad de versiones

- La búsqueda vectorial para Amazon DocumentDB solo está disponible en clústeres basados en instancias de Amazon DocumentDB 5.0 o superiores.

Vectores

- Amazon DocumentDB puede indexar vectores de hasta 2000 dimensiones. Sin embargo, se pueden almacenar hasta 16 000 dimensiones sin un índice.

Índices

- Para la creación de índices IVFFlat, la configuración recomendada para el parámetro de listas es el número de hasta 1 millón de documentos y documents/1000 para más de 1 millón de documentos. `sqrt(# of documents)` Debido al límite de memoria de trabajo, Amazon DocumentDB admite un determinado valor máximo del parámetro de listas en función del número de dimensiones. Como referencia, en la siguiente tabla se proporcionan los valores máximos del parámetro de listas para vectores de 500, 1000 y 2000 dimensiones:

Tipo de instancia	Listas con 500 dimensiones	Listas con 1000 dimensiones	Listas con 2000 dimensiones
t3.med	372	257	150
r5.l	915	741	511
r5.xl	1.393	1.196	901
r5.2xl	5.460	5.230	4.788
r5.4xl	7.842	7.599	7.138
r5.8xl	11.220	10.974	10.498
r5.12xl	13.774	13.526	13,044
r5.16xl	15.943	15.694	15.208

Tipo de instancia	Listas con 500 dimensiones	Listas con 1000 dimensiones	Listas con 2000 dimensiones
r5.24xl	19.585	19.335	18.845

- Los índices vectoriales no admiten otras opciones de índice, como `compound`, `sparse` o `partial`.
- La creación de índices paralelos no es compatible con el índice HNSW en Amazon DocumentDB 5.0.

Consulta vectorial

- Para una consulta de búsqueda vectorial, es importante refinar los parámetros, por ejemplo, `probes` o `efSearch`, para obtener resultados óptimos. Cuanto mayor sea el valor del parámetro `probes` o `efSearch`, mayor será la recuperación y menor será la velocidad. La configuración recomendada para iniciar el refinamiento del parámetro de sondas es `sqrt(# of lists)`.

Prácticas recomendadas

Descubra las prácticas recomendadas para trabajar con la búsqueda vectorial en Amazon DocumentDB. Esta sección se actualiza continuamente a medida que se identifican nuevas prácticas recomendadas.

- La creación del índice de Inverted File with Flat Compression (IVFFlat) implica agrupar en clústeres y organizar los puntos de datos en función de las similitudes. Para que un índice sea más eficaz, cargue algunos datos antes de crearlo.
- Para consultas de búsqueda vectorial, es importante refinar los parámetros, por ejemplo, `probes` o `efSearch`, para obtener resultados óptimos. Cuanto mayor sea el valor del parámetro `probes` o `efSearch`, mayor será la recuperación y menor será la velocidad. La configuración recomendada para empezar a refinar el parámetro `probes` es `sqrt(lists)`.

Recursos

- La búsqueda vectorial de Amazon DocumentDB ya está disponible de forma general en el blog AWS
- Ejemplo de código de búsqueda semántica

- Ejemplos de códigos de búsqueda vectorial de Amazon DocumentDB

Utilización AWS Lambda con flujos de cambio

Amazon DocumentDB está integrado AWS Lambda para que pueda usar las funciones de Lambda para procesar los registros de un flujo de cambios. La asignación de fuentes de eventos de Lambda es un recurso que se puede usar para invocar funciones de Lambda con el fin de procesar eventos de Amazon DocumentDB que no invocan Lambda directamente. Con el flujo de cambios de Amazon DocumentDB como origen de eventos, puede crear aplicaciones basadas en eventos que respondan a cambios en sus datos. Por ejemplo, puede utilizar funciones de Lambda para procesar documentos nuevos, realizar un seguimiento de actualizaciones de documentos existentes o registrar documentos eliminados.

Puede configurar una asignación de orígenes de eventos para enviar registros desde su flujo de cambios de Amazon DocumentDB a una función de Lambda. Los eventos pueden enviarse de uno en uno o por lotes para mejorar la eficiencia y se procesan por orden. Puede configurar el comportamiento de procesamiento de su asignación de orígenes de eventos en función de una duración específica (de 0 a 300 segundos) o del recuento de registros por lotes (límite máximo de 10 000 registros). Puede crear varios mapeos de fuentes de eventos para procesar los mismos datos con distintas funciones de Lambda o para procesar distintos elementos de varios flujos con una sola función.

Sin embargo, si la función devuelve un error, Lambda vuelve a intentar ejecutar el lote hasta que se procese correctamente. En caso de que los eventos del flujo de cambios hayan caducado, Lambda deshabilitará la asignación de orígenes de eventos. En este caso, puede crear una nueva asignación de orígenes de eventos y configurarla con la posición inicial que elija. Las asignaciones de orígenes de eventos Lambda procesan los eventos al menos una vez debido a la naturaleza distribuida de los sondeadores. Como resultado, la función de Lambda puede recibir eventos duplicados en situaciones excepcionales. Siga las prácticas recomendadas para trabajar con AWS Lambda funciones y cree funciones idempotentes para evitar problemas relacionados con la duplicación de eventos. Para obtener más información, consulte Uso AWS Lambda console con Amazon DocumentDB en la AWS Lambda Guía para desarrolladores.

Como práctica recomendada de rendimiento, la función de Lambda debe ser de corta duración. Para evitar introducir retrasos innecesarios en el procesamiento, tampoco debe ejecutar una lógica compleja. Para un flujo de alta velocidad en concreto, es mejor desencadenar un flujo de trabajo asíncrono de funciones de posprocesamiento que funciones Lambda sincrónicas de larga duración.

Para obtener más información al respecto AWS Lambda, consulte la Guía para AWS Lambda desarrolladores.

Limitaciones

Las siguientes son las limitaciones que hay que tener en cuenta al trabajar con Amazon DocumentDB y AWS Lambda:

- AWS Lambda actualmente solo se admite en Amazon DocumentDB 4.0 y 5.0.
- AWS Lambda actualmente no se admite en clústeres elásticos ni en clústeres globales.
- AWS Lambda los tamaños de carga útil no pueden superar los 6 MB. Para obtener más información sobre los tamaños de los lotes de Lambda, consulte “Comportamiento de procesamiento por lotes” en la sección Asignación de origen de eventos de Lambda de la Guía para desarrolladores de AWS Lambda .

Compatibilidad de Amazon DocumentDB con MongoDB

Amazon DocumentDB admite la compatibilidad con MongoDB, incluidos MongoDB 4.0, MongoDB 5.0 y MongoDB 8.0. La compatibilidad con MongoDB significa que la gran mayoría de las aplicaciones, controladores y herramientas que ya utiliza en la actualidad con sus bases de datos de MongoDB se pueden utilizar con Amazon DocumentDB con pocos o ningún cambio. En esta sección se describe todo lo que necesita saber sobre la compatibilidad de Amazon DocumentDB con MongoDB, incluyendo las nuevas capacidades y características, los primeros pasos, las rutas de migración y las diferencias funcionales.

Temas

- Compatibilidad con MongoDB 8.0
- Compatibilidad con MongoDB 5.0
- Compatibilidad con MongoDB 4.0
- Diferencias funcionales: Amazon DocumentDB y MongoDB
- API, operaciones y tipos de datos de MongoDB admitidos en Amazon DocumentDB

Compatibilidad con MongoDB 8.0

Temas

- Novedades de Amazon DocumentDB 8.0
- Comience con Amazon DocumentDB 8.0
- Actualice o migre a Amazon DocumentDB 5.0 u 8.0
- Diferencias funcionales

Novedades de Amazon DocumentDB 8.0

Amazon DocumentDB 8.0 mejora el rendimiento de las consultas hasta 7 veces y mejora la relación de compresión hasta 5 veces, lo que le permite crear aplicaciones de alto rendimiento a un costo menor. Amazon DocumentDB 8.0 es totalmente compatible con los protocolos cableados de MongoDB 8.0. El siguiente resumen presenta algunas de las principales características que se introdujeron en Amazon DocumentDB 8.0. Para ver una lista completa de nuevas capacidades, consulte Notas de la versión.

- Proporciona compatibilidad con MongoDB 8.0 al añadir soporte para los controladores de API de MongoDB 8.0. También es compatible con las aplicaciones que se crean con las versiones 6.0 y 7.0 de la API de MongoDB.
- El nuevo planificador de consultas de Amazon DocumentDB 8.0 amplía las mejoras de rendimiento a los operadores de la fase de agregación, además de admitir la optimización de las canalizaciones de agregación y distintos comandos.
- Amazon DocumentDB 8.0 admite la intercalación y las vistas.
- Ofrece 6 nuevas etapas de agregación: $replaceWith, $VectorSearch, $merge, $set, $unset, $bucket y 3 nuevos operadores de agregación: $pow, $rand y $dateTrunc.
- Una nueva versión del índice de texto: el índice de texto v2 de Amazon DocumentDB 8.0 presenta tokens adicionales que mejoran las capacidades de búsqueda de texto.
- Gracias a la creación de índices vectoriales en paralelo, Amazon DocumentDB 8.0 reduce el tiempo de creación de índices hasta 30 veces.
- Se aumentó el límite de almacenamiento a 256 TiB.

Comience con Amazon DocumentDB 8.0

Para empezar a utilizar Amazon DocumentDB 8.0, consulte la. Introducción a Amazon DocumentDB Puede crear un nuevo clúster de Amazon DocumentDB 8.0 con el AWS SDK, Consola de administración de AWS AWS CLI o. CloudFormation Al conectarse a Amazon DocumentDB, es necesario que utilice un controlador o una utilidad de MongoDB que sea compatible con MongoDB 5.0 o superior.

> **Note**
>
> Cuando utilice el AWS SDK, o AWS CLI CloudFormation, la versión del motor será la 5.0.0 de forma predeterminada. Debe especificar explícitamente el parámetro `engineVersion = 8.0.0` para crear un nuevo clúster de Amazon DocumentDB 8.0 o `engineVersion = 4.0.0` para crear un nuevo clúster de Amazon DocumentDB 4.0 o `engineVersion = 3.6.0` para crear un nuevo clúster de Amazon DocumentDB 3.6. Para un clúster de Amazon DocumentDB determinado, puede determinar la versión del clúster mediante la consola de administración de Amazon DocumentDB `describe-db-clusters` o la AWS CLI consola de administración de Amazon DocumentDB para ver el número de versión del motor de un clúster determinado.

Amazon DocumentDB 5.0 y 8.0 es compatible con los procesadores `r8g` Graviton4 y Graviton2 de Amazon EC2, así como los tipos de `t4.medium` instancia para sus clústeres, `r6g` y está disponible en todos los formatos compatibles (consulte). Regiones de AWS instancias Para obtener más información acerca del precio, consulte Amazon DocumentDB (with MongoDB compatibility) Pricing.

Actualice o migre a Amazon DocumentDB 5.0 u 8.0

Puede migrar de MongoDB 3.6 o MongoDB 4.0 a Amazon DocumentDB 5.0 o Amazon DocumentDB 8.0 mediante utilidades como mongodump, mongorestore, mongoimport y mongoexport. AWS DMS Para Actualización del clúster de Amazon DocumentDB mediante AWS Database Migration Service obtener instrucciones sobre la migración, consulte.

Diferencias funcionales

Diferencias funcionales entre Amazon DocumentDB 5.0 y 8.0

Con el lanzamiento de Amazon DocumentDB 8.0, existen diferencias funcionales entre Amazon DocumentDB 5.0 y Amazon DocumentDB 8.0:

- Planner v1 es el planificador de consultas predeterminado en Amazon DocumentDB 5.0, mientras que el Planner v3, de mayor rendimiento, es el predeterminado en Amazon DocumentDB 8.0.
- Las nuevas funciones de Amazon DocumentDB 8.0, como las vistas, la intercalación y operadores como $merge, solo son compatibles con Planner v3.
- La compresión está activada de forma predeterminada en Amazon DocumentDB 8.0 y está configurada para usar el algoritmo Zstandard. Además, en Amazon DocumentDB 8.0, la opción «habilitada» ya no es válida; puede seleccionar entre Zstd, LZ4 y ninguna.
- Amazon DocumentDB 8.0 solo admite TLS 1.2 y versiones posteriores para las conexiones a bases de datos.

> **Important**
>
> Amazon DocumentDB 8.0 requiere TLS 1.2 o superior. Ya no se admiten TLS 1.0 y TLS 1.1.

Compatibilidad con MongoDB 5.0

Temas

- Novedades de Amazon DocumentDB 5.0
- Novedades de Amazon DocumentDB 5.0.1
- Introducción a Amazon DocumentDB 5.0
- Actualice o migre a Amazon DocumentDB 5.0
- Diferencias funcionales

Novedades de Amazon DocumentDB 5.0

Amazon DocumentDB 5.0 presenta nuevas características y capacidades que incluyen límites de almacenamiento y cifrado a nivel de campo del lado del cliente. El siguiente resumen presenta algunas de las principales características que se introdujeron en Amazon DocumentDB 5.0. Para ver una lista completa de nuevas capacidades, consulte Notas de la versión.

- Se aumentó el límite de almacenamiento a 128 TiB para los clústeres de Amazon DocumentDB basados en instancias y para los clústeres elásticos basados en particiones.
- Se presentó el motor Amazon DocumentDB 5.0 (versión 3.0.775)
  - Compatibilidad con los controladores API de MongoDB 5.0
  - Soporte para el cifrado a nivel de Client-side campo (FLE). Ahora puede cifrar los campos del lado del cliente antes de escribir los datos en el clúster de Amazon DocumentDB. Para obtener más información, consulte Client-side cifrado a nivel de campo.
  - Nuevos operadores de agregación: `$dateAdd`, `$dateSubtract`
  - Compatibilidad con índices con operador `$elemMatch`. Como resultado, las consultas que se realicen con `$elemMatch`, darán lugar a escaneos del índice.

Amazon DocumentDB no es compatible con todas las características de MongoDB 5.0. Cuando creamos Amazon DocumentDB 5.0, partimos de las características y capacidades que nuestros clientes nos pedían más. Seguiremos añadiendo capacidades adicionales a MongoDB 5.0 en función de lo que nos pidan los clientes. Para ver la lista más reciente de las API compatibles, consulteAPI, operaciones y tipos de datos de MongoDB admitidos en Amazon DocumentDB.

Novedades de Amazon DocumentDB 5.0.1

Amazon DocumentDB 5.0.1 es la primera versión secundaria de la versión principal de Amazon DocumentDB 5.0. Las versiones secundarias incluyen mejoras como nuevas funciones, correcciones de seguridad y correcciones de errores. Puedes actualizar a la 5.0.1 desde la 5.0.0 mediante una

actualización de versión secundaria. Para obtener más información, consulte Actualización de la versión secundaria de Amazon DocumentDB.

- Capacidades de agregación mejoradas con nuevos operadores (`$rand`,, `$pow$dateToParts`,`$dateFromParts`).
- Métrica de conexiones activas para monitorear las instancias. Para obtener más información, consulte Cuotas de instancias.
- Las métricas de rendimiento granulares a nivel CloudWatch de comando incluyen las operaciones CRUD (`findinsert`,`findAndModify`,`update`,`delete`,`aggregate`,,`count`,`distinct`,`commitTransaction` `getMoreabortTransaction`, la exposición `AvgDuration` y `MaxConcurrent` las dimensiones para mejorar la supervisión operativa. `P100Duration`
- Amazon DocumentDB 5.0.1 solo admite TLS 1.2 y versiones posteriores para las conexiones a bases de datos.
- Se agregaron `gcRuntimeStats` y dividieron las métricas del nivel de recolección `storageSegmentExtended` en `storageSegmentBase` y dentro `collstats` para obtener más información sobre el estado general del proceso de recolección de basura. `documentFragmentStats`

Introducción a Amazon DocumentDB 5.0

Para empezar a utilizar Amazon DocumentDB 5.0, consulte. Introducción a Amazon DocumentDB Puede crear un nuevo clúster de Amazon DocumentDB 5.0 utilizando el AWS SDK, Consola de administración de AWS AWS CLI o. CloudFormation Al conectarse a Amazon DocumentDB, es necesario que utilice un controlador o una utilidad de MongoDB que sea compatible con MongoDB 5.0 o superior.

> **Note**
>
> Cuando utilice el AWS SDK o AWS CLI CloudFormation, la versión del motor será la 5.0.0 de forma predeterminada. Debe especificar de forma explícita el parámetro `engineVersion = 4.0.0` para crear un nuevo clúster de Amazon DocumentDB 4.0 o `engineVersion = 3.6.0` para crear un nuevo clúster de Amazon DocumentDB 3.6. Para un clúster de Amazon DocumentDB determinado, puede determinar la versión del clúster mediante la función de llamada `describe-db-clusters` o AWS CLI utilizar la consola de administración de Amazon DocumentDB para ver el número de versión del motor de un clúster determinado.

Amazon DocumentDB 5.0 y 8.0 es compatible con los procesadores `r8g` Graviton4 y Graviton2 de Amazon EC2, así como los tipos de `t4.medium` instancia para sus clústeres, `r6g` y está disponible en todos los formatos compatibles (consulte). Regiones de AWS instancias Para obtener más información acerca del precio, consulte Amazon DocumentDB (with MongoDB compatibility) Pricing.

Actualice o migre a Amazon DocumentDB 5.0

Puede migrar de MongoDB 3.6 o MongoDB 4.0 a Amazon DocumentDB 5.0 mediante utilidades como mongodump, mongorestore AWS DMS, mongoimport y mongoexport. Para obtener Actualización del clúster de Amazon DocumentDB mediante AWS Database Migration Service instrucciones sobre la migración, consulte.

Diferencias funcionales

Diferencias funcionales entre Amazon DocumentDB 4.0 y 5.0

Con el lanzamiento de Amazon DocumentDB 5.0, existen diferencias funcionales entre Amazon DocumentDB 4.0 y Amazon DocumentDB 5.0:

- La función de copia de seguridad integrada ahora es compatible con `serverStatus`. Acción: los desarrolladores y las aplicaciones con función de backup pueden recopilar estadísticas sobre el estado del clúster de Amazon DocumentDB.
- El campo `SecondaryDelaySecs` reemplaza a `slaveDelay` en la salida `replSetGetConfig`.
- El comando hello reemplaza a `isMaster` - hello devuelve un documento que describe la función de un clúster de Amazon DocumentDB.
- Amazon DocumentDB 5.0 ahora es compatible con escaneos de índices con el operador `$elemMatch` en el primer nivel de anidación. Los escaneos de índices son compatibles cuando el filtro para solo consultas tiene un nivel de filtro de `$elemMatch`, pero no son compatibles si se incluye una consulta de `$elemMatch` anidada.

  Por ejemplo, en Amazon DocumentDB 5.0, si usted incluye el operador `$elemMatch` en el nivel anidado, no devolverá un valor como lo hace en 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 proyección “$” de Amazon DocumentDB 4.0 devuelve todos los documentos con todos los campos. Con Amazon DocumentDB 5.0, el comando find con una proyección “$” devuelve los documentos que coinciden con el parámetro de consulta y contienen solo el campo que coincide con la proyección “$”.
- En Amazon DocumentDB 5.0, los comandos find con parámetros de consulta `$regex` y `$options` los comandos devuelven un error: “No se pueden establecer opciones en ambos`$regex` y `$options`”.
- Con Amazon DocumentDB 5.0, `$indexOfCP` ahora devuelve “-1” cuando:
  - la subcadena no se encuentra en la expresión de cadena, o
  - el inicio es un número mayor que el final, o
  - el inicio es un número mayor que la longitud en bytes de la cadena.
- En Amazon DocumentDB 4.0, `$indexOfCP` devuelve “0” cuando la posición inicial es un número mayor que el final o que la longitud en bytes de la cadena.
- Con Amazon DocumentDB 5.0, las operaciones de proyección en `_id fields`, por ejemplo `{"_id.nestedField" : 1}`, devuelven documentos que solo incluyen el campo proyectado. En cambio, en Amazon DocumentDB 4.0, los comandos de proyección de campos anidados no filtran ningún documento.

Compatibilidad con MongoDB 4.0

Temas

- Características de Amazon DocumentDB 4.0
- Introducción a Amazon DocumentDB 4.0
- Actualización o migración a Amazon DocumentDB 4.0
- Diferencias funcionales

Características de Amazon DocumentDB 4.0

Amazon DocumentDB 4.0 introdujo muchas características y capacidades nuevas que incluían transacciones ACID y mejoras en los flujos de cambios. El siguiente resumen presenta algunas de las principales características que se introdujeron en Amazon DocumentDB 4.0. Para ver una lista completa de capacidades, consulte Notas de la versión.

- Transacciones ACID: Amazon DocumentDB ahora es compatible con la capacidad de realizar transacciones en varios documentos, estados de cuenta, colecciones y bases de datos. Las transacciones simplifican el desarrollo de aplicaciones al permitirle realizar operaciones atómicas, consistentes, aisladas y duraderas (ACID) en uno o más documentos dentro de un clúster de Amazon DocumentDB. Para obtener más información, consulte Transacciones en Amazon DocumentDB.
- Flujos de cambios: ahora puede abrir un flujo de cambios a nivel de clúster (`client.watch()` o `mongo.watch()`) y la base de datos (`db.watch()`), puede especificar un `startAtOperationTime` para abrir el cursor de flujo de cambios y, por último, puede ampliar el período de retención del flujo de cambios a 7 días (anteriormente 24 horas). Para obtener más información, consulte Uso de flujos de cambios con Amazon DocumentDB.
- AWS Database Migration Service(AWS DMS): Ahora puede utilizarla AWS DMS para migrar sus cargas de trabajo de MongoDB 4.0 a Amazon DocumentDB. AWS DMS ahora admite una fuente de MongoDB 4.0, un destino de Amazon DocumentDB 4.0 y una fuente de Amazon DocumentDB 3.6 para realizar actualizaciones entre Amazon DocumentDB 3.6 y 4.0. Para obtener más información, consulte la Documentación de AWS DMS.
- Rendimiento e indexación: puede utilizar un índice con `$lookup`, buscar consultas con una proyección que contenga un campo o un campo y el campo `_id` puede servir directamente desde el índice sin necesidad de leer la colección (consulta cubierta), capacidad de `hint()` con

`findAndModify`, optimizaciones del rendimiento para `$addToSet` y mejoras para reducir el tamaño general de los índices. Para obtener más información, consulte Notas de la versión.

- Operadores: Amazon DocumentDB 4.0 ahora es compatible varios operadores de agregación nuevos: `$ifNull`, `$replaceRoot`, `$setIsSubset`, `$setIntersection`, `$setUnion`, `$setEquals`. Puede ver todas las API, operaciones y tipos de datos de MongoDB compatibles en API, operaciones y tipos de datos de MongoDB admitidos en Amazon DocumentDB.
- Control de acceso basado en funciones (RBAC): con los comandos `ListCollection` y `ListDatabase`, ahora puede usar opcionalmente los parámetros `authorizedCollections` y `authorizedDatabases` para permitir a los usuarios enlistar las colecciones y bases de datos a las que tienen permiso de acceso sin necesitar las funciones `listCollections` y `listDatabase`, respectivamente. También puede eliminar sus propios cursores sin necesitar la función `KillCursor`.

Amazon DocumentDB no es compatible con todas las características de MongoDB 4.0. Cuando creamos Amazon DocumentDB 4.0, partimos de las características y capacidades que nuestros clientes nos pedían más. Seguiremos añadiendo capacidades adicionales a MongoDB 4.0 en función de lo que nos pidan los clientes. Por ejemplo, Amazon DocumentDB 4.0 actualmente no es compatible con los operadores de conversión de tipos ni con los operadores de cadenas que se introdujeron en MongoDB 4.0. Para obtener la lista más reciente de las API compatibles, consulte. API, operaciones y tipos de datos de MongoDB admitidos en Amazon DocumentDB

Introducción a Amazon DocumentDB 4.0

Para empezar a utilizar Amazon DocumentDB 4.0, consulte. Introducción a Amazon DocumentDB Puede crear un nuevo clúster de Amazon DocumentDB 4.0 utilizando el AWS SDK, Consola de administración de AWS AWS CLI o. CloudFormation Al conectarse a Amazon DocumentDB, es necesario que utilice un controlador o una utilidad de MongoDB que sea compatible con MongoDB 4.0 o superior.

> **Note**
>
> Cuando utilice el AWS SDK o AWS CLI CloudFormation, la versión del motor será la 5.0.0 de forma predeterminada. Debe especificar de forma explícita el parámetro `engineVersion = 4.0.0` para crear un nuevo clúster de Amazon DocumentDB 4.0 o `engineVersion = 3.6.0` para crear un nuevo clúster de Amazon DocumentDB 3.6. Para un clúster de Amazon DocumentDB determinado, puede determinar la versión del clúster mediante la función

de llamada `describe-db-clusters` o AWS CLI utilizar la consola de administración de Amazon DocumentDB para ver el número de versión del motor de un clúster determinado.

Amazon DocumentDB 4.0 admite los tipos de `t4g.medium` instancia `r5r6g`,`t3.medium`, y para sus clústeres, y está disponible en todos los tipos compatibles. Regiones de AWS La utilización de Amazon DocumentDB 4.0 no implica costos adicionales. Para obtener más información acerca del precio, consulte Amazon DocumentDB (with MongoDB compatibility) Pricing.

Actualización o migración a Amazon DocumentDB 4.0

Puede migrar de MongoDB 3.6 o MongoDB 4.0 a Amazon DocumentDB 4.0 mediante utilidades como mongodump, mongorestore AWS DMS, mongoimport y mongoexport. De forma similar, puede utilizar las mismas herramientas para actualizar de Amazon DocumentDB 3.6 a Amazon DocumentDB 4.0. Para obtener Actualización del clúster de Amazon DocumentDB mediante AWS Database Migration Service instrucciones sobre la migración, consulte.

Diferencias funcionales

Diferencias funcionales entre Amazon DocumentDB 3.6 y 4.0

Con el lanzamiento de Amazon DocumentDB 4.0, existen diferencias funcionales entre Amazon DocumentDB 3.6 y Amazon DocumentDB 4.0:

- Proyección para documentos anidados: Amazon DocumentDB 3.6 considera el primer campo de un documento anidado al aplicar una proyección. Sin embargo, Amazon DocumentDB 4.0 analizará los subdocumentos y aplicará la proyección a cada uno de ellos. Por ejemplo: si la proyección lo es `"a.b.c": 1`, el comportamiento en ambas versiones es idéntico. Sin embargo, si la proyección es `{a:{b:{c:1}}}`, Amazon DocumentDB 3.6 solo aplicará la proyección a “a” y no a “b” o “c”.
- Comportamiento de **`minKey`**, **`maxKey`**: con Amazon DocumentDB 4.0, el comportamiento de `{x:{$gt:MaxKey}}` no devuelve nada y para `{x:{$lt:MaxKey}}`, lo devuelve todo.
- Diferencias en la comparación de documentos: la comparación de valores numéricos de distintos tipos (doble, int, long) en subdocumentos (p. ej., b en `{"_id" :1, "a" :{"b":1}}`) ahora proporciona un resultado coherente en todos los tipos de datos numéricos y para cada nivel de un documento.

Diferencias funcionales entre Amazon DocumentDB 4.0 y MongoDB 4.0

A continuación, se muestran las diferencias funcionales entre Amazon DocumentDB 4.0 y MongoDB 4.0.

- Búsqueda de una clave vacía en la ruta: cuando una colección contiene un documento con una clave vacía dentro de la matriz (por ejemplo `{"x" : [ { "" : 10 }, { "b" : 20 } ]}`) y cuando la clave utilizada en la consulta termina en una cadena vacía (por ejemplo `x.`), Amazon DocumentDB devolverá ese documento, ya que recorre todos los documentos de la matriz, mientras que MongoDB no lo devolverá.
- **`$setOnInsert`** junto con **`$`** en la ruta: el operador de campo `$setOnInsert` no funcionará en combinación con $ en la ruta en Amazon DocumentDB, lo que también es coherente con MongoDB 4.0.

Diferencias funcionales: Amazon DocumentDB y MongoDB

A continuación se explican las diferencias funcionales entre Amazon DocumentDB (compatible con MongoDB) y MongoDB.

Temas

- Beneficios funcionales de Amazon DocumentDB
- Diferencias funcionales actualizadas
- Diferencias funcionales con MongoDB

Beneficios funcionales de Amazon DocumentDB

Transacciones implícitas

En Amazon DocumentDB, todas las declaraciones de CRUD (`findAndModify`, `update`, `insert` y `delete`) garantizan la atomicidad y la coherencia, incluso en operaciones que modifican varios documentos. Con el lanzamiento de Amazon DocumentDB 4.0, ahora se admiten transacciones explícitas que proporcionan propiedades ACID para operaciones de varios estados de cuenta y cobros. Para obtener más información sobre el uso de transacciones en Amazon DocumentDB, consulteTransacciones en Amazon DocumentDB.

A continuación se muestran ejemplos de operaciones en Amazon DocumentDB que modifican varios documentos que cumplen los comportamientos de atomicidad y coherencia.

```
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 }]
})
```

Las operaciones individuales que componen operaciones en bloque tales como `updateMany` y `deleteMany` son atómicas, pero la operación en bloque en su conjunto no es atómica. Por ejemplo, la operación `insertMany` en su conjunto es atómica si las operaciones de inserción individuales se ejecutan correctamente sin errores. Si se detecta algún error en una operación `insertMany`,

cada instrucción de inserción individual dentro de la operación `insertMany` se ejecutará como una operación atómica. Si necesita propiedades ACID para operaciones de `insertMany`, `updateMany` y `deleteMany`, se recomienda utilizar una transacción.

Diferencias funcionales actualizadas

Amazon DocumentDB continúa mejorando la compatibilidad con MongoDB al trabajar a partir de las capacidades que nuestros clientes nos piden que creemos. Esta sección contiene las diferencias funcionales que hemos eliminado en Amazon DocumentDB para facilitar las migraciones y la creación de aplicaciones para nuestros clientes.

Temas

- Indexación de matrices
- Multi-key índices
- Caracteres nulos en cadenas
- Role-based control de acceso
- indexación $regex
- Proyección para documentos anidados

Indexación de matrices

A partir del 23 de abril de 2020, Amazon DocumentDB admite la capacidad de indexar matrices mayores de 2048 bytes. El límite para un elemento individual en una matriz se mantiene en 2048 bytes, lo que es coherente con MongoDB.

Si crea un nuevo índice, no se necesita ninguna acción para aprovechar la funcionalidad mejorada. Si tiene un índice existente, puede aprovechar la funcionalidad mejorada borrando el índice y después volviéndolo a crear. La versión del índice actual con las capacidades mejoradas es `"v" : 3`.

> **Note**
>
> En el caso de los clústeres de producción, eliminar el índice podría repercutir en el rendimiento de las aplicaciones. Realice primero una prueba y proceda con precaución al realizar cambios en un sistema de producción. Además, el tiempo necesario para volver a crear el índice depende del tamaño total de los datos de la colección.

Puede consultar la versión de los índices mediante el siguiente comando.

```
db.collection.getIndexes()
```

La salida de esta operación será similar a lo que se indica a continuación. En esta salida, la versión del índice es `"v" : 3`, que es la versión de índice más actual.

```
[
    {
        "v" : 3,
        "key" : {
        "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.test"
    }
]
```

Multi-key índices

A partir del 23 de abril de 2020, Amazon DocumentDB admite la capacidad de crear un índice compuesto por varias claves en la misma matriz.

Si crea un nuevo índice, no se necesita ninguna acción para aprovechar la funcionalidad mejorada. Si tiene un índice existente, puede aprovechar la funcionalidad mejorada borrando el índice y después volviéndolo a crear. La versión del índice actual con las capacidades mejoradas es `"v" : 3`.

> **Note**
>
> En el caso de los clústeres de producción, eliminar el índice puede afectar al rendimiento de las aplicaciones. Realice primero una prueba y proceda con precaución al realizar cambios en un sistema de producción. Además, el tiempo necesario para volver a crear el índice depende del tamaño total de los datos de la colección.

Puede consultar la versión de los índices mediante el siguiente comando.

```
db.collection.getIndexes()
```

La salida de esta operación será similar a lo que se indica a continuación. En esta salida, la versión del índice es `"v" : 3`, que es la versión de índice más actual.

```
[
    {
        "v" : 3,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.test"
    }
]
```

Caracteres nulos en cadenas

A partir del 22 de junio de 2020, Amazon DocumentDB admite ahora caracteres nulos (`'\0'`) en cadenas.

Role-based control de acceso

A partir del 26 de marzo de 2020, Amazon DocumentDB admite el control de acceso basado en roles (RBAC) para roles integrados. Para obtener más información, consulte Role-Based Control de acceso.

indexación $regex

A partir del 22 de junio de 2020, Amazon DocumentDB admite la capacidad de los operadores `$regex` de utilizar un índice.

Para utilizar un índice con el operador `$regex`, debe utilizar el comando `hint()`. Al utilizar `hint()`, debe especificar el nombre del campo en el que está aplicando `$regex`. Por ejemplo, si tiene un índice en el campo `product` con el nombre de índice p_1, `db.foo.find({product: /^x.*/}).hint({product:1})` utilizará el índice p_1, pero `db.foo.find({product: /^x.*/}).hint("p_1")` no utilizará el índice. Puede comprobar si se elige un índice mediante el comando `explain()` o haciendo uso del generador de perfiles para registrar consultas lentas. Por ejemplo, `db.foo.find({product: /^x.*/}).hint("p_1").explain()`.

> **Note**
>
> Solo puede usarse un índice cada vez con el método `hint()`.

El uso de un índice en una consulta `$regex` está optimizado para consultas regex que usan un prefijo y no especifican las opciones `i`, `m` o `o` de regex.

Al utilizar un índice con `$regex`, se recomienda crear un índice en campos altamente selectivos donde el número de valores duplicados sea inferior al 1 % del número total de documentos de la colección. Por ejemplo, si la colección cuenta con 100 000 documentos, solo cree índices en campos donde el mismo valor se produzca 1000 veces o menos.

Proyección para documentos anidados

Existe una diferencia funcional con el operador `$project` entre Amazon DocumentDB y MongoDB en la versión 3.6 que se ha resuelto en Amazon DocumentDB 4.0, pero seguirá sin ser compatible con Amazon DocumentDB 3.6.

Amazon DocumentDB 3.6 solo tiene en cuenta el primer campo de un documento anidado al aplicar una proyección, mientras que MongoDB 3.6 analizará los subdocumentos y aplicará la proyección también a cada subdocumento.

Por ejemplo: si la proyección es `"a.b.c": 1`, se comportará como se esperaba tanto en Amazon DocumentDB como en MongoDB. Sin embargo, si la proyección es `{a:{b:{c:1}}}`, Amazon DocumentDB 3.6 solo aplicará la proyección a `a` y no a `b` o `c`. En Amazon DocumentDB 4.0, la proyección `{a:{b:{c:1}}}` se aplicará a `a`, `b` y `c`.

Diferencias funcionales con MongoDB

Temas

- Operador $VectorSearch
- OpCountersCommand
- Bases de datos de administración y colecciones
- CursorMaxTimeMS
- explain()
- Compilaciones de índices

- Búsqueda con una clave vacía en la ruta
- API, operaciones y tipos de datos de MongoDB
- utilidades mongodump y mongorestore
- Ordenación de los resultados
- Reintento de las escrituras
- Índices dispersos
- Usar $elemMatch dentro de una expresión $all
- Dólar($) y punto(.) en los nombres de campos
- $lookup
- $natural y clasificación inversa

Operador $VectorSearch

Amazon DocumentDB no admite `$vectorSearch` como operador independiente. En su lugar, admitimos `vectorSearch` dentro del operador `$search`. Para obtener más información, consulte Búsqueda vectorial para Amazon DocumentDB.

`OpCountersCommand`

El comportamiento de `OpCountersCommand` de Amazon DocumentDB se desvía de `opcounters.command` de MongoDB de la siguiente manera:

- `opcounters.command` de MongoDB cuenta todos los comandos excepto el de insertar, actualizar y eliminar, mientras que `OpCountersCommand` de Amazon DocumentDB también excluye el comando `find`.
- Amazon DocumentDB tiene en cuenta los comandos internos para `OpCountersCommand`.

Bases de datos de administración y colecciones

Amazon DocumentDB no admite la base de datos de administración o local ni las colecciones `system.*` o `startup_log` de MongoDB, respectivamente.

`CursorMaxTimeMS`

En Amazon DocumentDB, `cursor.maxTimeMS` restablece el contador de cada solicitud de `getMore`. Por lo tanto, si se especifica un `maxTimeMS` de 3000 MS, la consulta tarda 2800 MS

y cada solicitud de `getMore` posterior tarda 300 MS, por lo que el cursor no agotará el tiempo de espera. El tiempo de espera del cursor solo se agotará cuando una sola operación, ya sea la consulta o una solicitud de `getMore` individual, dure más que el `maxTimeMS` especificado. Además, el barrido que comprueba el tiempo de ejecución del cursor funciona con una granularidad de cinco (5) minutos.

explain()

Amazon DocumentDB emula las API 3.6, 4.0, 5.0 y 8.0 de MongoDB en un motor de base de datos diseñado específicamente que utiliza un sistema de almacenamiento distribuido, tolerante a errores y con recuperación automática. Como resultado, los planes de consulta y la salida de `explain()` pueden diferir entre Amazon DocumentDB y MongoDB. Los clientes que deseen controlar su plan de consulta pueden utilizar el operador `$hint` para aplicar la selección de un índice preferido.

Compilaciones de índices

Amazon DocumentDB solo permite una operación de creación de índice en una colección al mismo tiempo. Ya sea en primer plano o en segundo plano. Si operaciones, tales como `createIndex()` o `dropIndex()`, se producen en la misma colección cuando una operación de creación de índice está actualmente en curso, se producirá un error en la operación que se ha intentado realizar recientemente.

De forma predeterminada, las compilaciones de índices en Amazon DocumentDB y MongoDB versión 4.0 se producen en primer plano. La versión 4.2 y posteriores de MongoDB ignoran la opción de creación de índices en segundo plano si se especifica en createIndexes o sus asistentes de intérprete de comandos `createIndex()` y `createIndexes()`.

Un índice de tiempo de vida (TTL) empieza a marcar los documentos como caducados en cuanto se completa la operación de creación del índice.

Búsqueda con una clave vacía en la ruta

Si busca con una clave que incluye una cadena vacía como parte de la ruta (por ejemplo, `x.`, `x..b`) y el objeto tiene una ruta de clave de cadena vacía (por ejemplo, `{"x" : [ { "" : 10 }, { "b" : 20 } ]}`) dentro de una matriz, Amazon DocumentDB devolverá resultados diferentes a los que arrojaría si usted ejecutara la misma búsqueda en MongoDB.

En MongoDB, la búsqueda de rutas con clave vacía dentro de una matriz funciona tal y como se esperaba cuando la clave de cadena vacía no está al final de la búsqueda de rutas. Sin embargo, cuando la clave de cadena vacía está al final de la búsqueda de rutas, no busca en la matriz.

Sin embargo, en Amazon DocumentDB, solo se lee el primer elemento de la matriz, ya que `getArrayIndexFromKeyString` convierte una cadena vacía en `0`, por lo que se trata a la búsqueda de claves de cadena como a una búsqueda de índice de matriz.

API, operaciones y tipos de datos de MongoDB

Amazon DocumentDB es compatible con las API 3.6, 4.0, 5.0 y 8.0 de MongoDB. Para ver la lista actualizada de funcionalidades admitidas, consulte API, operaciones y tipos de datos de MongoDB admitidos en Amazon DocumentDB.

`utilidades mongodump y mongorestore`

Amazon DocumentDB no admite una base de datos de administración y, por lo tanto, no vuelca ni restaura la base de datos de administración cuando se usan las utilidades `mongodump` o `mongorestore`. Al crear una nueva base de datos en Amazon DocumentDB mediante `mongorestore`, debe volver a crear los roles de usuario además de la operación de restauración.

> **Requisito de versión de MongoDB Database Tools**
>
> Utilice las herramientas de base de datos de MongoDB hasta la versión 100.11.0, incluida, para Amazon DocumentDB. Para descargarlas, consulte las versiones de las herramientas de base de datos de MongoDB en el sitio web de MongoDB.

Ordenación de los resultados

Amazon DocumentDB no garantiza la ordenación implícita de los conjuntos de resultados. Para garantizar la ordenación de un conjunto de resultados, especifique explícitamente un criterio de ordenación utilizando `sort()`.

En el siguiente ejemplo, se ordenan los elementos de la colección de inventario en orden descendente en función del campo "stock".

```
db.inventory.find().sort({ stock: -1 })
```

Cuando se utiliza la etapa de agregación de `$sort`, el orden de clasificación no se conserva a menos que la etapa `$sort` sea la última etapa del proceso de agregación. Cuando se utiliza la etapa de agregación de `$sort` en combinación con la etapa de agregación de `$group`, la etapa

de agregación de `$sort` solo se aplica a los acumuladores de `$first` y `$last`. En Amazon DocumentDB 4.0, se agregó compatibilidad con `$push` para respetar el orden de clasificación de la etapa de `$sort` anterior.

Reintento de las escrituras

La mayoría de los MongoDB-compatible controladores actuales permiten volver a intentarlo de forma predeterminada. Amazon DocumentDB admite escrituras que se pueden volver a intentar a partir de la versión 8.0.2 del motor. En las versiones anteriores del motor, no se admitían las escrituras reintentables y debe deshabilitarlas para evitar errores.

En las versiones del motor anteriores a la 8.0.2, si se habilitan las escrituras reintentables, se generaba un mensaje de error similar al siguiente.

```
{"ok":0,"errmsg":"Retryable writes are not supported","code":301,"name":"MongoError"}
```

Las escrituras reintentables se pueden deshabilitar mediante la cadena de conexión (por ejemplo,`MongoClient("mongodb://my.mongodb.cluster/db?retryWrites=false")`) o el argumento de palabra clave del MongoClient constructor (por ejemplo,). `MongoClient("mongodb://my.mongodb.cluster/db", retryWrites=False)`

Para obtener más información sobre las escrituras que se pueden volver a intentar en la versión 8.0.2 y posteriores del motor, consulte. Escrituras que se pueden volver a intentar en Amazon DocumentDB

Índices dispersos

Para utilizar un índice disperso que haya creado en una consulta, debe utilizar la cláusula `$exists` en los campos incluidos en el índice. Si omite `$exists`, Amazon DocumentDB no utilizará el índice disperso.

A continuación se muestra un ejemplo.

```
db.inventory.count({ "stock": { $exists: true }})
```

Para índices dispersos de varias claves, Amazon DocumentDB no admite una restricción de clave única si la búsqueda de un documento da como resultado un conjunto de valores y solo falta un subconjunto de los campos indexados. Por ejemplo, `createIndex({"a.b" : 1 }, { unique : true, sparse :true })` no se admite con la entrada `"a" : [ { "b" : 2 }, { "c" : 1 } ]`, ya que `"a.c"` se almacena en el índice.

Usar `$elemMatch dentro de una expresión $all`

Actualmente, Amazon DocumentDB no admite el uso del operador `$elemMatch` dentro de una expresión `$all`. Como solución alternativa, puede usar el operador `$and` con `$elemMatch` de la siguiente manera.

Operación original:

```
db.col.find({
  qty: {
    $all: [
      { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } },
      { "$elemMatch": { num: 40, size: "XL" } }
    ]
  }
})
```

Operación actualizada:

```
db.col.find({
  $and: [
    { qty: { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } } },
    { qty: { "$elemMatch": { qty: 40, size: "XL" } } }
  ]
})
```

Dólar($) y punto(.) en los nombres de campos

Amazon DocumentDB no admite la consulta de campos con prefijo dólar($) en $in, $nin y $all en objetos anidados. Por ejemplo, la siguiente consulta no es válida en Amazon DocumentDB.

```
coll.find({"field": {"$all": [{ "$a": 1 }]}})
```

`$lookup`

Amazon DocumentDB admite la posibilidad de realizar coincidencias equitativas (por ejemplo, la unión externa a la izquierda) y también admite subconsultas no correlacionadas. A partir de la versión 8.0.2, Amazon DocumentDB también admite subconsultas correlacionadas, incluida la sintaxis concisa de las subconsultas correlacionadas. En una subconsulta correlacionada, solo un

subconjunto de etapas de agregación que evalúan las expresiones puede hacer referencia a las variables. `let` Para ver la lista de las etapas admitidas y las restricciones relacionadas, consulte. Etapas compatibles en las subconsultas correlacionadas En las versiones anteriores a la 8.0.2, no se admiten las subconsultas correlacionadas.

Utilización de un índice con $lookup

Ahora puede utilizar un índice con el operador de la etapa de `$lookup`. Según su caso de uso, existen varios algoritmos de indexación que puede utilizar para optimizar el rendimiento. En esta sección se explican los diferentes algoritmos de indexación para `$lookup` y se le ayuda a elegir el mejor para su carga de trabajo.

De forma predeterminada, Amazon DocumentDB utilizará el algoritmo hash cuando se utilice `allowDiskUse:false` y se realizará la fusión de clasificación cuando se use `allowDiskUse:true`.

> **Note**
>
> Actualmente, la opción `allowDiskUse` no es compatible con el comando `find`. La opción solo se admite como parte de la agregación. Utilice el marco de agregación `allowDiskUse:true` para gestionar consultas de gran tamaño que puedan superar los límites de memoria.

En algunos casos de uso, puede ser preferible obligar al optimizador de consultas a utilizar un algoritmo diferente. A continuación se muestran los diferentes algoritmos de indexación que puede utilizar el operador de agregación de `$lookup`:

- Bucle anidado: un plan de bucles anidados suele ser beneficioso para una carga de trabajo si la colección externa es inferior a 1 GB y el campo de la colección externa tiene un índice. Si se utiliza el algoritmo de bucle anidado, el plan explicativo mostrará la etapa como `NESTED_LOOP_LOOKUP`.
- Fusión y ordenación: un plan de fusión y ordenación suele ser beneficioso para una carga de trabajo si la colección externa no tiene un índice en el campo utilizado en la búsqueda y el conjunto de datos de trabajo no cabe en la memoria. Si se utiliza el algoritmo de fusión y ordenación, el plan explicativo mostrará la etapa como `SORT_LOOKUP`.
- Hash: un plan de hash suele ser beneficioso para una carga de trabajo si la colección externa ocupa <1 GB y el conjunto de datos de trabajo cabe en la memoria. Si se utiliza el algoritmo de hash, el plan explicativo mostrará la etapa como `HASH_LOOKUP`.

Puede identificar el algoritmo de indexación que se utiliza para el operador de `$lookup` si utiliza `explain` en la consulta. A continuación se muestra un ejemplo:

```
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
}
```

Como alternativa al uso del método de `explain()`, puede usar el generador de perfiles para revisar el algoritmo que se utiliza al usar el operador de `$lookup`. Para obtener más información sobre el generador de perfiles, consulteCreación de perfiles de operaciones en Amazon DocumentDB.

Uso de un PlanHint

Si desea obligar al optimizador de consultas a utilizar un algoritmo de indexación diferente con `$lookup`, puede utilizar un `planHint`. Para ello, utilice el comentario en las opciones de la etapa de agregación para forzar un plan diferente. A continuación, se muestra un ejemplo de la sintaxis del comentario:

```
comment : {
    comment :  "<string>",
    lookupStage : { planHint : "SORT" | "HASH" | "NESTED_LOOP" }
}
```

A continuación, se muestra un ejemplo del uso de `planHint` para obligar al optimizador de consultas a utilizar el algoritmo de indexación `HASH`:

```
db.foo.aggregate(
   [
      {
         $lookup:
            {
               from: "foo",
               localField: "_id",
               foreignField: "_id",
               as: "joined"
```

```
            },
      }
   ]
),
{
   comment : "{ \"lookupStage\" : { \"planHint\": \"HASH\" }}"
```

Para probar qué algoritmo se adapta mejor a su carga de trabajo, puede utilizar el parámetro `executionStats` del método `explain` para medir el tiempo de ejecución de la etapa de `$lookup` y, al mismo tiempo, modificar el algoritmo de indexación (es decir, `HASH`/`SORT`/`NESTED_LOOP`).

El siguiente ejemplo muestra cómo utilizar `executionStats` para medir el tiempo de ejecución de la etapa de `$lookup` mediante el algoritmo de `SORT`.

```
db.foo.explain("executionStats").aggregate(
   [
      {
         $lookup:
            {
               from: "foo",
               localField: "_id",
               foreignField: "_id",
               as: "joined"
            },
      }
   ]
),
{
   comment : "{ \"lookupStage\" : { \"planHint\": \"SORT\" }}"
```

`$natural` y clasificación inversa

Amazon DocumentDB admite `$natural` solamente para escaneos de colecciones reenviados. Los escaneos de colecciones inversos (`{$natural: -1}`) conducirán a un `MongoServerError`.

API, operaciones y tipos de datos de MongoDB admitidos en Amazon DocumentDB

Amazon DocumentDB (compatible con MongoDB) es un servicio de base de datos de documentos rápido, completamente administrado, de alta disponibilidad y escala ajustable que admite cargas

de trabajo de MongoDB. Amazon DocumentDB es compatible con las API 3.6, 4.0, 5.0 y 8.0 de MongoDB. Esta sección muestra la funcionalidad admitida. Para obtener soporte para el uso de las API y los controladores de MongoDB, consulte los foros de la comunidad de MongoDB. Para obtener asistencia sobre el uso del servicio Amazon DocumentDB, póngase en contacto con el equipo de soporte correspondiente. AWS Para conocer las diferencias funcionales entre Amazon DocumentDB y MongoDB, consulte. Diferencias funcionales: Amazon DocumentDB y MongoDB

Los comandos y operadores de MongoDB que son solo para uso interno o que no se aplican a un servicio completamente administrado no se admiten ni están incluidos en la lista de funcionalidades admitidas.

Hemos agregado más de 50 capacidades adicionales desde el lanzamiento y seguiremos trabajando con versiones anteriores de nuestros clientes para ofrecer las capacidades que necesitan.
Para obtener información sobre los lanzamientos más recientes, consulte Anuncios de Amazon DocumentDB.

Si hay una característica que no es compatible que desea que creemos, háganoslo saber enviando un correo electrónico con su ID de cuenta, las características solicitadas y el caso de uso al equipo de servicio de Amazon DocumentDB.

Temas

- Comandos de la base de datos
- Operadores de consulta y proyección
- Operadores de actualización
- Geospatial (Geoespacial)
- Métodos de cursor
- Operadores de canalización de agregación
- Tipos de datos
- Índices

Comandos de la base de datos

Temas

- Comandos administrativos
- Agregación

- Autenticación
- Comandos de diagnóstico
- Operaciones de consulta y escritura
- Comandos para la administración de roles
- Comandos de sesiones
- Administración de usuarios
- Comandos de partición

Comandos administrativos

Comando	3.6	4.0	5.0	8.0	Clúster elástico
Colecciones limitadas	No	No	No	No	No
clonar CollectionAsCapped	No	No	No	No	No
collMod	Parcial	Parcial	Parcial	Parcial	Parcial
CollMod: caducarAfterSeconds	Sí	Sí	Sí	Sí	Sí
CollMod: oculto	No	No	Sí (5.0.2+)	Sí (8.0.1+)	No
convertirToCapped	No	No	No	No	No
copydb	No	No	No	No	No
crear	Sí	Sí	Sí	Sí	Sí
createView	No	No	No	Sí	No

Comando	3.6	4.0	5.0	8.0	Clúster elástico
createIndexes	Sí	Sí	Sí	Sí	Sí
currentOp	Sí	Sí	Sí	Sí	Sí
drop	Sí	Sí	Sí	Sí	Sí
dropDatabase	Sí	Sí	Sí	Sí	Sí
dropIndexes	Sí	Sí	Sí	Sí	Sí
filemd5	No	No	No	No	No
obtenerAuditConfig	No	Sí	Sí	Sí	No
killCursors	Sí	Sí	Sí	Sí	Sí
killOp	Sí	Sí	Sí	Sí	Sí
listCollections*	Sí	Sí	Sí	Sí	Sí
listDatabases	Sí	Sí	Sí	Sí	Sí
listIndexes	Sí	Sí	Sí	Sí	Sí
reIndex	No	No	Sí	Sí	No
renameCollection	Sí	Sí	Sí	Sí	No
conjuntoAuditConfig	No	Sí	Sí	Sí	No

* No se admite la clave `type` de la opción de filtro.

Agregación

Comando	3.6	4.0	5.0	8.0	Clúster elástico
aggregate	Sí	Sí	Sí	Sí	Sí
count	Sí	Sí	Sí	Sí	Sí
distinct	Sí	Sí	Sí	Sí	Sí
mapReduce	No	No	No	Sí	No

Autenticación

Comando	3.6	4.0	5.0	8.0	Clúster elástico
authenticate	Sí	Sí	Sí	Sí	Sí
logout	Sí	Sí	Sí	Sí	Sí

Comandos de diagnóstico

Comando	3.6	4.0	5.0	8.0	Clúster elástico
buildInfo	Sí	Sí	Sí	Sí	Sí
collStats	Sí	Sí	Sí	Sí	Sí
conn PoolStats	No	No	No	No	No
connectio nStatus	Sí	Sí	Sí	Sí	Sí

Comando	3.6	4.0	5.0	8.0	Clúster elástico
dataSize	Sí	Sí	Sí	Sí	Sí
dbHash	No	No	No	No	No
dbStats	Sí	Sí	Sí	Sí	Sí
explain	Sí	Sí	Sí	Sí	Sí
explain: executionStats	Sí	Sí	Sí	Sí	Sí
Características de	No	No	No	No	No
hostInfo	Sí	Sí	Sí	Sí	Sí
listCommands	Sí	Sí	Sí	Sí	Sí
planificarCacheClearFilters	No	No	Sí	Sí	No
planificarCacheListFilters	No	No	Sí	Sí	No
planificarCacheSetFilter	No	No	Sí	Sí	No
profiler	Sí	Sí	Sí	Sí	No
serverStatus	Sí	Sí	Sí	Sí	Sí
top	Sí	Sí	Sí	Sí	Sí

Operaciones de consulta y escritura

Comando	3.6	4.0	5.0	8.0	Clúster elástico
Change streams	Sí	Sí	Sí	Sí	No
eliminar	Sí	Sí	Sí	Sí	Sí
find	Sí	Sí	Sí	Sí	Sí
encontrar AndModify	Sí	Sí	Sí	Sí	Sí
conseguir LastError	No	No	No	No	No
getMore	Sí	Sí	Sí	Sí	Sí
conseguir PrevError	No	No	No	No	No
GridFS	Sí	Sí	Sí	Sí	No
inserción	Sí	Sí	Sí	Sí	Sí
paralelo Collectio nScan	No	No	No	No	No
resetError	No	No	No	No	No
actualización	Sí	Sí	Sí	Sí	Sí
ReplaceOne	Sí	Sí	Sí	Sí	Sí

Comandos para la administración de roles

C	3.6	4.0	5.0	8.0	Clúster elástico
c e	Sí	Sí	Sí	Sí	No
c A r s	Sí	Sí	Sí	Sí	No
d	Sí	Sí	Sí	Sí	No
c F l	Sí	Sí	Sí	Sí	No
r F F	Sí	Sí	Sí	Sí	No
r F s	Sí	Sí	Sí	Sí	No
r	Sí	Sí	Sí	Sí	No
u e	Sí	Sí	Sí	Sí	No

Comandos de sesiones

Comando	3.6	4.0	5.0	8.0	Clúster elástico
Abortar transacción	No	Sí	Sí	Sí	No
commitTransaction	No	Sí	Sí	Sí	No
Finalizar sesiones	No	No	No	No	No
matar AllSessions	No	Sí	Sí	Sí	No
matar AllSessionsByPattern	No	No	No	No	No
Mata a Sessions	No	Sí	Sí	Sí	No
Actualizar sesiones	No	No	No	No	No
StartSession	No	Sí	Sí	Sí	No

Administración de usuarios

Comando	3.6	4.0	5.0	8.0	Clúster elástico
createUser	Sí	Sí	Sí	Sí	Sí

Comando	3.6	4.0	5.0	8.0	Clúster elástico
caer AllUsersFromDatabase	Sí	Sí	Sí	Sí	Sí
dropUser	Sí	Sí	Sí	Sí	Sí
conceder RolesToUser	Sí	Sí	Sí	Sí	Sí
revocar RolesFrom User	Sí	Sí	Sí	Sí	Sí
updateUser	Sí	Sí	Sí	Sí	Sí
usersInfo	Sí	Sí	Sí	Sí	Sí

Comandos de partición

Comando	Clúster elástico
abortar ReshardCollection	No
Añadir partición	No
añadir ShardToZone	No
balanceador CollectionStatus	No
Balancer Start	No
Estado del equilibrador	No
BalancerStop	No
comprobar ShardingIndex	No

Comando	Clúster elástico
claro JumboFlag	No
cleanupOrphaned	No
limpieza ReshardCollection	No
comprometerse ReshardCollection	No
Habilitar Sharding	Sí
rubor RouterConfig	No
conseguir ShardMap	No
conseguir ShardVersion	No
isdbgrid	No
ListShards	No
Clave mediana	No
Mueva Chunk	No
Mueva el modo principal	No
MergeChunks	No
refinar CollectionShardKey	No
Elimina Hard	No
eliminar ShardFromZone	No
Colección ReShard	No
conjunto AllowMigrations	No
conjunto ShardVersion	No

Comando	Clúster elástico
Colección Shard	Sí
Estado de partición	No
dividir	No
Vector dividido	No
UnsetSharding	No
actualizar ZoneKeyRange	No

Operadores de consulta y proyección

Temas

- Operadores de matrices
- Operadores Bitwise
- Operador de comentarios
- Operadores de comparación
- Operadores de elementos
- Operadores de consulta de evaluación
- Logical operators (Operadores lógicos)
- Operadores de proyección

Operadores de matrices

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$all	Sí	Sí	Sí	Sí	Sí
$elemMatch	Sí	Sí	Sí	Sí	Sí
$size	Sí	Sí	Sí	Sí	Sí

Operadores Bitwise

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$ bits AllSet	Sí	Sí	Sí	Sí	Sí
$ bits AnySet	Sí	Sí	Sí	Sí	Sí
$ bits AllClear	Sí	Sí	Sí	Sí	Sí
$ bits AnyClear	Sí	Sí	Sí	Sí	Sí

Operador de comentarios

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$comment	Sí	Sí	Sí	Sí	Sí

Operadores de comparación

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$eq	Sí	Sí	Sí	Sí	Sí
$gt	Sí	Sí	Sí	Sí	Sí
$gte	Sí	Sí	Sí	Sí	Sí
$in	Sí	Sí	Sí	Sí	Sí
$lt	Sí	Sí	Sí	Sí	Sí
$lte	Sí	Sí	Sí	Sí	Sí

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$ne	Sí	Sí	Sí	Sí	Sí
$nin	Sí	Sí	Sí	Sí	Sí

Operadores de elementos

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$exists	Sí	Sí	Sí	Sí	Sí
$type	Sí	Sí	Sí	Sí	Sí

Operadores de consulta de evaluación

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$expr	No	Sí	Sí	Sí	No
$jsonSchema	No	Sí	Sí	Sí	No
$mod	Sí	Sí	Sí	Sí	Sí
$regex	Sí	Sí	Sí	Sí	Sí
$text	No	No	Sí	Sí	No
$where	No	No	No	No	No

Logical operators (Operadores lógicos)

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$and	Sí	Sí	Sí	Sí	Sí
$nor	Sí	Sí	Sí	Sí	Sí
$not	Sí	Sí	Sí	Sí	Sí
$or	Sí	Sí	Sí	Sí	Sí

Operadores de proyección

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$	Sí	Sí	Sí	Sí	Sí
$elemMatch	Sí	Sí	Sí	Sí	Sí
$meta	No	No	Sí	Sí	No
$slice	Sí	Sí	Sí	Sí	Sí

Operadores de actualización

Temas

- Operadores de matrices
- Operadores Bitwise
- Operadores de campo
- Modificadores de actualización

Operadores de matrices

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$	Sí	Sí	Sí	Sí	Sí
$[]	Sí	Sí	Sí	Sí	Sí
$ [<identificador>]	Sí	Sí	Sí	Sí	Sí
$addToSet	Sí	Sí	Sí	Sí	Sí
$pop	Sí	Sí	Sí	Sí	Sí
$pullAll	Sí	Sí	Sí	Sí	Sí
$pull	Sí	Sí	Sí	Sí	Sí
$push	Sí	Sí	Sí	Sí	Sí

Operadores Bitwise

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$bit	Sí	Sí	Sí	Sí	Sí

Operadores de campo

Operador	3.6	4.0	5.0	8.0	Clúster elástico
$currentDate	Sí	Sí	Sí	Sí	Sí
$inc	Sí	Sí	Sí	Sí	Sí

Operador	3.6	4.0	5.0	8.0	Clúster elástico
$max	Sí	Sí	Sí	Sí	Sí
$min	Sí	Sí	Sí	Sí	Sí
$mul	Sí	Sí	Sí	Sí	Sí
$rename	Sí	Sí	Sí	Sí	Sí
$set	Sí	Sí	Sí	Sí	Sí
$setOnInsert	Sí	Sí	Sí	Sí	Sí
$unset	Sí	Sí	Sí	Sí	Sí

Modificadores de actualización

Operador	3.6	4.0	5.0	8.0	Clúster elástico
$each	Sí	Sí	Sí	Sí	Sí
$position	Sí	Sí	Sí	Sí	Sí
$slice	Sí	Sí	Sí	Sí	Sí
$sort	Sí	Sí	Sí	Sí	Sí

Geospatial (Geoespacial)

Especificadores de geometría

Selectores de consultas	3.6	4.0	5.0	8.0	Clúster elástico
$box	No	No	No	No	No

Selectores de consultas	3.6	4.0	5.0	8.0	Clúster elástico
$center	No	No	No	No	No
$centerSphere	No	No	No	No	No
$geometry	Sí	Sí	Sí	Sí	Sí
$maxDistance	Sí	Sí	Sí	Sí	Sí
$minDistance	Sí	Sí	Sí	Sí	Sí
$nearSphere	Sí	Sí	Sí	Sí	Sí
$polygon	No	No	No	No	No
$uniqueDocs	No	No	No	No	No

Selectores de consultas

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$geoIntersects	Sí	Sí	Sí	Sí	Sí
$geoWithin	Sí	Sí	Sí	Sí	Sí
$near	Sí	Sí	Sí	Sí	Sí
$nearSphere	Sí	Sí	Sí	Sí	Sí
$polygon	No	No	No	No	No
$uniqueDocs	No	No	No	No	No

Métodos de cursor

Comando	3.6	4.0	5.0	8.0	Clúster elástico
cursor.batchSize()	Sí	Sí	Sí	Sí	Sí
cursor.close()	Sí	Sí	Sí	Sí	Sí
cursor.collation()	No	No	No	Sí	No
cursor.comment()	Sí	Sí	Sí	Sí	Sí
cursor.count()	Sí	Sí	Sí	Sí	Sí
cursor.explain()	Sí	Sí	Sí	Sí	No
cursor.forEach()	Sí	Sí	Sí	Sí	Sí
cursor.hasNext()	Sí	Sí	Sí	Sí	Sí
cursor.hint()	Sí	Sí	Sí	Sí	Sí*
cursor.isClosed()	Sí	Sí	Sí	Sí	Sí
cursor.isExhausted()	Sí	Sí	Sí	Sí	No
cursor.itcount()	Sí	Sí	Sí	Sí	No
cursor.limit()	Sí	Sí	Sí	Sí	No

Comando	3.6	4.0	5.0	8.0	Clúster elástico
cursor.map()	Sí	Sí	Sí	Sí	No
cursor.max()	No	No	Sí (5.0.2+)	Sí (8.0.1+)	No
cursor.maxScan()	Sí	Sí	Sí	Sí	No
cursor.maxTimeMS()	Sí	Sí	Sí	Sí	No
cursor.min()	No	No	Sí (5.0.2+)	Sí (8.0.1+)	No
cursor.next()	Sí	Sí	Sí	Sí	Sí
cursor.no () CursorTimeout	No	No	No	No	No
cursor.objs () LeftInBatch	Sí	Sí	Sí	Sí	No
cursor.pretty()	Sí	Sí	Sí	Sí	No
cursor.readConcern()	Sí	Sí	Sí	Sí	No
cursor.readPref()	Sí	Sí	Sí	Sí	No
cursor.returnKey()	No	No	No	No	No
cursor.showRecordId ()	No	No	No	No	No
cursor.size()	Sí	Sí	Sí	Sí	No

Comando	3.6	4.0	5.0	8.0	Clúster elástico
cursor.skip()	Sí	Sí	Sí	Sí	No
cursor.sort()	Sí	Sí	Sí	Sí	No
cursor.tailable()	No	No	No	No	No
cursor.toArray()	Sí	Sí	Sí	Sí	No

* El índice `hint` es compatible con las expresiones de índice. Por ejemplo, `db.foo.find().hint({x:1})`.

Los `cursor.max()` métodos `cursor.min()` y requieren un nombre para el índice `hint` que se va a utilizar. Ambos métodos deben especificar los mismos campos y no se pueden combinar en una `$natural` clasificación. Solo se admiten en el `find` comando.

Operadores de canalización de agregación

Temas

- Expresiones de acumulación
- Operadores aritméticos
- Operadores de matrices
- Operadores booleanos
- Operadores de comparación
- Operadores de expresiones condicionales
- Operador de tipos de datos
- Operador de tamaño de datos
- Operadores de fechas
- Operador literal
- Operador de combinación
- Operador natural

- Operadores de establecimiento
- Operadores de etapa
- Operadores de cadena
- Variables del sistema
- Operador de búsqueda de texto
- Operadores de conversión de tipos
- Operadores de variables
- Operadores de trigonometría
- Operadores bit a bit (agregación)
- Operadores de marca de tiempo
- Operadores misceláneos

Expresiones de acumulación

Expression	3.6	4.0	5.0	8.0	Clúster elástico
$ acumulador	-	-	No	No	No
$añadir ToSet	Sí	Sí	Sí	Sí	Sí
$avg	Sí	Sí	Sí	Sí	Sí
$inferior	-	-	-	Sí (8.0.1+)	No
$ a N	-	-	-	Sí (8.0.1+)	No
$count	-	-	No	Sí (8.0.1+)	No
$covarian cePop	No	No	No	Sí (8.0.2+)	No
$covarian ceSamp	No	No	No	Sí (8.0.2+)	No

Expression	3.6	4.0	5.0	8.0	Clúster elástico
$denseRank	No	No	No	Sí (8.0.2+)	No
$derivative	No	No	No	No	No
$document Number	No	No	No	Sí (8.0.2+)	No
$ exp MovingAvg	No	No	No	No	No
$first	Sí	Sí	Sí	Sí	Sí
$primeroN	-	-	-	Sí (8.0.1+)	No
$integral	No	No	No	No	No
$last	Sí	Sí	Sí	Sí	Sí
$Último N	-	-	-	Sí (8.0.1+)	No
$ LinearFill	No	No	No	No	No
$locf	No	No	No	No	No
$max	Sí	Sí	Sí	Sí	Sí
$ máx. N	-	-	-	Sí (8.0.1+)	No
$ mediana	-	-	-	Sí (8.0.1+)	No
$min	Sí	Sí	Sí	Sí	Sí
$ mN	-	-	-	Sí (8.0.1+)	No
percentil $	-	-	-	Sí (8.0.1+)	No
$push	Sí	Sí	Sí	Sí	Sí
$rank	No	No	No	Sí (8.0.2+)	No

Expression	3.6	4.0	5.0	8.0	Clúster elástico
$shift	No	No	No	No	No
$ estándar DevPop	No	No	No	Sí (8.0.1+)	No
$ estándar DevSamp	No	No	No	Sí (8.0.1+)	No
$sum	Sí	Sí	Sí	Sí	Sí
$superior	-	-	-	Sí (8.0.1+)	No
$ a PN	-	-	-	Sí (8.0.1+)	No

Operadores aritméticos

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$abs	Sí	Sí	Sí	Sí	Sí
$add	Sí	Sí	Sí	Sí	Sí
$ceil	No	Sí	Sí	Sí	Sí
$divide	Sí	Sí	Sí	Sí	Sí
$exp	No	Sí	Sí	Sí	Sí
$floor	No	Sí	Sí	Sí	Sí
$ln	No	Sí	Sí	Sí	Sí
$log	No	Sí	Sí	Sí	Sí
$log10	No	Sí	Sí	Sí	Sí

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$mod	Sí	Sí	Sí	Sí	Sí
$multiply	Sí	Sí	Sí	Sí	Sí
$pow	No	No	Sí (5.0.1+)	Sí	No
$round	-	-	No	Sí (8.0.1+)	No
$sqrt	No	Sí	Sí	Sí	Sí
$subtract	Sí	Sí	Sí	Sí	Sí
$trunc	No	No	No	Sí (8.0.1+)	No

Operadores de matrices

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$matrizElemAt	Sí	Sí	Sí	Sí	Sí
$matrizToObject	Sí	Sí	Sí	Sí	Sí
$concatArrays	Sí	Sí	Sí	Sí	Sí
$filter	Sí	Sí	Sí	Sí	Sí
$first	-	-	Sí	Sí	No
$in	Sí	Sí	Sí	Sí	Sí
$índiceOfArray	Sí	Sí	Sí	Sí	Sí

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$isArray	Sí	Sí	Sí	Sí	Sí
$last	-	-	Sí	Sí	No
$objetoToArray	Sí	Sí	Sí	Sí	Sí
$range	Sí	Sí	Sí	Sí	Sí
$reverseArray	Sí	Sí	Sí	Sí	Sí
$SortArray	No	No	No	Sí	No
$reduce	Sí	Sí	Sí	Sí	Sí
$size	Sí	Sí	Sí	Sí	Sí
$slice	Sí	Sí	Sí	Sí	Sí
$zip	Sí	Sí	Sí	Sí	Sí

Operadores booleanos

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$and	Sí	Sí	Sí	Sí	Sí
$not	Sí	Sí	Sí	Sí	Sí
$or	Sí	Sí	Sí	Sí	Sí

Operadores de comparación

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$cmp	Sí	Sí	Sí	Sí	Sí
$eq	Sí	Sí	Sí	Sí	Sí
$gt	Sí	Sí	Sí	Sí	Sí
$gte	Sí	Sí	Sí	Sí	Sí
$lt	Sí	Sí	Sí	Sí	Sí
$lte	Sí	Sí	Sí	Sí	Sí
$ne	Sí	Sí	Sí	Sí	Sí

Operadores de expresiones condicionales

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$cond	Sí	Sí	Sí	Sí	Sí
$ifNull	Sí	Sí	Sí	Sí	Sí
$switch	No	Sí	Sí	Sí	No

Operador de tipos de datos

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$type	Sí	Sí	Sí	Sí	Sí

Operador de tamaño de datos

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$binarySize	-	-	No	Sí (8.0.1+)	No
$bsonSize	-	-	No	Sí (8.0.1+)	No

Operadores de fechas

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$dateAdd	No	No	Sí	Sí	Sí
$dateDiff	-	-	Sí	Sí	No
$ fecha FromParts	No	No	Sí (5.0.1+)	No	No
$ fecha FromString	Sí	Sí	Sí	Sí	Sí
$dateSubtract	No	No	Sí	Sí	Sí
$ fecha ToParts	No	No	Sí (5.0.1+)	No	No
$ fecha ToString	Sí	Sí	Sí	Sí	Sí
$dateTrunc	-	-	No	Sí	No
$día OfMonth	Sí	Sí	Sí	Sí	Sí
$día OfWeek	Sí	Sí	Sí	Sí	Sí

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$día OfYear	Sí	Sí	Sí	Sí	Sí
$hour	Sí	Sí	Sí	Sí	Sí
$iso DayOfWeek	Sí	Sí	Sí	Sí	Sí
$isoWeek	Sí	Sí	Sí	Sí	Sí
$iso WeekYear	Sí	Sí	Sí	Sí	Sí
$millisecond	Sí	Sí	Sí	Sí	Sí
$minute	Sí	Sí	Sí	Sí	Sí
$month	Sí	Sí	Sí	Sí	Sí
$second	Sí	Sí	Sí	Sí	Sí
$week	Sí	Sí	Sí	Sí	Sí
$year	Sí	Sí	Sí	Sí	Sí

Operador literal

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$literal	Sí	Sí	Sí	Sí	Sí

Operador de combinación

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$mergeObjects	Sí	Sí	Sí	Sí	Sí

Operador natural

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$natural	Sí	Sí	Sí	Sí	Sí

Operadores de establecimiento

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$todosElementsTrue	No	Sí	Sí	Sí	Sí
$cualquierElementTrue	No	Sí	Sí	Sí	Sí
$setDifference	No	Sí	Sí	Sí	Sí
$setEquals	Sí	Sí	Sí	Sí	Sí
$setIntersection	Sí	Sí	Sí	Sí	Sí

Comando	3.6	4.0	5.0	8.0	Clúster elástico
conjunto $ IsSubset	Sí	Sí	Sí	Sí	Sí
$setUnion	Sí	Sí	Sí	Sí	Sí

Operadores de etapa

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$addFields	Sí	Sí	Sí	Sí	Sí
$bucket	No	No	No	Sí	No
$bucketAuto	No	No	No	Sí (8.0.2+)	No
$changeSt ream	Sí	Sí	Sí	Sí	No
$cambiar StreamSpl itLargeEvent	No	No	No	Sí (8.0.2+)	No
$collStats	No	Sí	Sí	Sí	No
$count	Sí	Sí	Sí	Sí	Sí
$currentOp	Sí	Sí	Sí	Sí	Sí
$densificar	No	No	No	No	No
$facet	No	No	No	Sí (8.0.2+)	No
$fill	No	No	No	No	No
$geoNear	Sí	Sí	Sí	Sí	Sí

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$graphLookup	No	No	No	Sí	No
$group	Sí	Sí	Sí	Sí	Sí
$indexStats	Sí	Sí	Sí	Sí	No
$limit	Sí	Sí	Sí	Sí	Sí
$listaLocalSessions	No	No	No	No	No
$listSessions	No	No	No	No	No
$lookup	Sí	Sí	Sí	Sí	Sí
$match	Sí	Sí	Sí	Sí	Sí
$merge	-	-	No	Sí	No
$out	Sí	Sí	Sí	Sí	No
$planCacheStats	-	-	No	No	No
$project	Sí	Sí	Sí	Sí	Sí
$redact	Sí	Sí	Sí	Sí	Sí
$replaceRoot	Sí	Sí	Sí	Sí	Sí
$sample	Sí	Sí	Sí	Sí	Sí
$set	-	-	No	Sí	No

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$conjuntoWindowFields	-	-	No	Sí (8.0.2+)	No
$skip	Sí	Sí	Sí	Sí	Sí
$sort	Sí	Sí	Sí	Sí	Sí
$ordenarByCount	No	No	No	Sí (8.0.1+)	No
$UnionWith	-	-	No	No	No
$unset	-	-	No	Sí	No
$unwind	Sí	Sí	Sí	Sí	Sí
$ Reemplazar con	No	No	No	Sí	No
$ VectorSearch	No	No	No	Sí	No
$listaSearchIndexes	-	-	-	Sí (8.0.1+)	No

Operadores de cadena

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$concat	Sí	Sí	Sí	Sí	Sí

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$índice OfBytes	Sí	Sí	Sí	Sí	Sí
$indexOfCP	Sí	Sí	Sí	Sí	Sí
$ltrim	No	Sí	Sí	Sí	No
$regexFind	-	-	Sí	Sí	No
$ expresión regular FindAll	-	-	Sí	Sí	No
$ RegexMatch	-	-	Sí	Sí	No
$ReplaceAll	-	-	Sí	Sí	No
$ ReplaceOne	-	-	Sí	Sí	No
$rtrim	No	Sí	Sí	Sí	No
$split	Sí	Sí	Sí	Sí	Sí
$strcasecmp	Sí	Sí	Sí	Sí	Sí
$star LenBytes	Sí	Sí	Sí	Sí	Sí
$strLenCP	Sí	Sí	Sí	Sí	Sí
$substr	Sí	Sí	Sí	Sí	Sí
$substrBytes	Sí	Sí	Sí	Sí	Sí
$substrCP	Sí	Sí	Sí	Sí	Sí

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$toLower	Sí	Sí	Sí	Sí	Sí
$toUpper	Sí	Sí	Sí	Sí	Sí
$trim	No	Sí	Sí	Sí	No

Variables del sistema

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$$CURRENT	No	No	No	No	No
$$DESCEND	Sí	Sí	Sí	Sí	Sí
$$KEEP	Sí	Sí	Sí	Sí	Sí
$$PRUNE	Sí	Sí	Sí	Sí	Sí
$$REMOVE	No	No	No	No	No
$ RAÍZ	Sí	Sí	Sí	Sí	Sí

Operador de búsqueda de texto

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$meta	No	No	Sí	Sí	No
$search	No	No	Sí	Sí	No

Operadores de conversión de tipos

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$convert	No	Sí	Sí	Sí	Sí
$es un número	-	-	No	Sí (8.0.1+)	No
$ a Bool	No	Sí	Sí	Sí	Sí
$toDate	No	Sí	Sí	Sí	Sí
$ a Decimal	No	Sí	Sí	Sí	Sí
$ a doble	No	Sí	Sí	Sí	Sí
$toInt	No	Sí	Sí	Sí	Sí
$ a Long	No	Sí	Sí	Sí	Sí
$ a ObjectId	No	Sí	Sí	Sí	Sí
$toString	No	Sí	Sí	Sí	Sí
$ a UUID	-	-	-	Sí (8.0.1+)	No

Operadores de variables

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$let	Sí	Sí	Sí	Sí	Sí
$map	Sí	Sí	Sí	Sí	Sí

Operadores de trigonometría

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$ acos	-	-	-	Sí (8.0.1+)	No
$ un cosh	-	-	-	Sí (8.0.1+)	No
$ ASIN	-	-	-	Sí (8.0.1+)	No
$asin	-	-	-	Sí (8.0.1+)	No
$ atán	-	-	-	Sí (8.0.1+)	No
$atan2	-	-	-	Sí (8.0.1+)	No
$atanh	-	-	-	Sí (8.0.1+)	No
$cos	-	-	-	Sí (8.0.1+)	No
$cosh	-	-	-	Sí (8.0.1+)	No
$ grados ToRadians	-	-	-	Sí (8.0.1+)	No
$ radianes ToDegrees	-	-	-	Sí (8.0.1+)	No
$sin	-	-	-	Sí (8.0.1+)	No
$sinh	-	-	-	Sí (8.0.1+)	No
$tan	-	-	-	Sí (8.0.1+)	No
$tanh	-	-	-	Sí (8.0.1+)	No

Operadores bit a bit (agregación)

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$ bit y	-	-	-	Sí (8.0.1+)	No
$bitNo	-	-	-	Sí (8.0.1+)	No
$ BIT o	-	-	-	Sí (8.0.1+)	No
$ bitX o	-	-	-	Sí (8.0.1+)	No

Operadores de marca de tiempo

Comando	3.6	4.0	5.0	8.0	Clúster elástico
Incremento de $ t	-	-	-	Sí (8.0.1+)	No
$tsSegundo	-	-	-	Sí (8.0.1+)	No

Operadores misceláneos

Comando	3.6	4.0	5.0	8.0	Clúster elástico
$getField	-	-	Sí	Sí	No
$rand	-	-	Sí (5.0.1+)	Sí	No
$sampleRate	-	-	No	Sí (8.0.1+)	No
$ sigmoide	-	-	-	Sí (8.0.1+)	No

Tipos de datos

Comando	3.6	4.0	5.0	8.0	Clúster elástico
32-bit Integer (int)	Sí	Sí	Sí	Sí	Sí
Entero de 64 bits (largo)	Sí	Sí	Sí	Sí	Sí
Matriz	Sí	Sí	Sí	Sí	Sí
Datos Binary	Sí	Sí	Sí	Sí	Sí
Booleano	Sí	Sí	Sí	Sí	Sí
Date	Sí	Sí	Sí	Sí	Sí
DBPointer	No	No	No	No	No
DBRefs	No	No	No	No	No
Decimal128	Sí	Sí	Sí	Sí	Sí
Double	Sí	Sí	Sí	Sí	Sí
JavaScript	No	No	No	No	No
JavaScript(con alcance)	No	No	No	No	No
MaxKey	Sí	Sí	Sí	Sí	Sí
MinKey	Sí	Sí	Sí	Sí	Sí
Nulo	Sí	Sí	Sí	Sí	Sí
Objeto	Sí	Sí	Sí	Sí	Sí

Comando	3.6	4.0	5.0	8.0	Clúster elástico
ObjectId	Sí	Sí	Sí	Sí	Sí
Expresión regular	Sí	Sí	Sí	Sí	Sí
Cadena	Sí	Sí	Sí	Sí	Sí
Símbolo	No	No	No	No	No
Timestamp	Sí	Sí	Sí	Sí	Sí
Sin definir	No	No	No	No	No

Índices y propiedades de índices

Temas

- Índices
- Propiedades de índices

Índices

Comando	3.6	4.0	5.0	8.0	Clúster elástico
2dsphere	Sí	Sí	Sí	Sí	Sí
Índice 2d	No	No	No	No	No
Índice compuesto	Sí	Sí	Sí	Sí	Sí
Índice hash	No	No	No	No	No

Comando	3.6	4.0	5.0	8.0	Clúster elástico
Índice de varias claves	Sí	Sí	Sí	Sí	Sí
Índice de campo único	Sí	Sí	Sí	Sí	Sí
Índice de texto	No	No	Sí	Sí	No
Comodín	No	No	No	No	No

Propiedades de índices

Comando	3.6	4.0	5.0	8.0	Clúster elástico
Introducción	Sí	Sí	Sí	Sí	Sí
No distingue entre mayúsculas y minúsculas	No	No	No	Sí	No
Oculto	No	No	No	Sí (8.0.1+)	No
Parcial	No	No	Sí	Sí	No
Sparse	Sí	Sí	Sí	Sí	Sí
Texto	No	No	Sí	Sí	No
TTL	Sí	Sí	Sí	Sí	Sí
Unique	Sí	Sí	Sí	Sí	Sí
Vector	No	No	Sí	Sí	No

Para obtener información detallada sobre operadores específicos de MongoDB, consulte los siguientes temas:

- Operadores de canalización de agregación
- Geospatial (Geoespacial)
- Operadores de proyección
- Operadores de actualización

Operadores de canalización de agregación

En esta sección se proporciona información detallada sobre los operadores de canalización de agregación compatibles con Amazon DocumentDB.

Temas

- $$DESCEND
- $$KEEP
- $$PRUNE
- $ RAÍZ
- $abs
- $ acos
- $ un cosh
- $add
- $añadir ToSet
- $addFields
- $todos ElementsTrue
- $and
- $cualquier ElementTrue
- $matriz ElemAt
- $matriz ToObject
- $ ASIN
- $asin
- $ atán

- $atan2
- $atanh
- $avg
- $binarySize
- $ bit y
- $bitNo
- $ BIT o
- $ bitX o
- $inferior
- $ a N
- $bsonSize
- $bucket
- $bucketAuto
- $ceil
- $changeStream
- $cambiar StreamSplitLargeEvent
- $cmp
- $collStats
- $concat
- $concatArrays
- $cond
- $convert
- $cos
- $cosh
- $count (etapa)
- $count (acumulador)
- $covariancePop
- $covarianceSamp
- $currentOp

- $dateAdd
- $dateDiff
- $ fecha FromParts
- $ fecha FromString
- $dateSubtract
- $ fecha ToParts
- $ fecha ToString
- $dateTrunc
- $día OfMonth
- $día OfWeek
- $día OfYear
- $ grados ToRadians
- $denseRank
- $divide
- $documentNumber
- $eq
- $exp
- $facet
- $filter
- $first
- $primeroN
- $floor
- $geoNear
- $graphLookup
- $group
- $gt
- $gte
- $hour
- $ifNull

- $in
- $índice OfArray
- $índice OfBytes
- $indexOfCP
- $indexStats
- $isArray
- $es un número
- $iso DayOfWeek
- $isoWeek
- $iso WeekYear
- $last
- $Último N
- $let
- $limit
- $lista SearchIndexes
- $literal
- $ln
- $log
- $log10
- $lt
- $lte
- $lookup
- $ltrim
- $map
- $match
- $max
- $ máx. N
- $ mediana
- $meta

- $merge
- $mergeObjects
- $millisecond
- $min
- $ mN
- $minute
- $mod
- $month
- $multiply
- $natural
- $ne
- $not
- $objeto ToArray
- $or
- $out
- percentil $
- $pow
- $push
- $project
- $ radianes ToDegrees
- $rand
- $range
- $rank
- $redact
- $reduce
- $regexFind
- $ expresión regular FindAll
- $ RegexMatch
- $ReplaceAll

- $ ReplaceOne
- $replaceRoot
- $ Reemplazar con
- $reverseArray
- $round
- $rtrim
- $sample
- $search
- $second
- $set
- $setDifference
- $setEquals
- $setIntersection
- conjunto $ IsSubset
- $setUnion
- $conjunto WindowFields
- $ sigmoide
- $sin
- $sinh
- $skip
- $slice
- $size
- $sort
- $SortArray
- $ordenar ByCount
- $split
- $sqrt
- $ estándar DevPop
- $ estándar DevSamp
- $star LenBytes

- $strLenCP
- $strcasecmp
- $substr
- $substrBytes
- $substrCP
- $subtract
- $sum
- $switch
- $tan
- $tanh
- $superior
- $ a PN
- $trunc
- Incremento de $ t
- $tsSegundo
- $ a Bool
- $toDate
- $ a Decimal
- $ a doble
- $toInt
- $ a Long
- $toLower
- $ a ObjectId
- $toString
- $toUpper
- $ a UUID
- $trim
- $type
- $unset
- $unwind

- $ VectorSearch
- $week
- $year
- $zip

$$DESCEND

El `$$DESCEND` operador de Amazon DocumentDB es un operador de matriz posicional especial que se utiliza en la `$redact` fase de canalización. Indica a la canalización de agregación que descienda al documento actual y procese todos los campos, independientemente de su nivel de anidación.

Cuando la `$redact` etapa se encuentre con el `$$DESCEND` operador, mantendrá visibles todos los campos del documento actual y los procesará más adelante. Esto resulta útil cuando se desea redactar o eliminar determinados campos de forma selectiva en función de una condición y, al mismo tiempo, conservar la estructura del documento.

Parámetros

Ninguna.

Ejemplo (MongoDB Shell)

En este ejemplo, utilizaremos la `$redact` etapa con el `$$DESCEND` operador para mostrar de forma selectiva los documentos en los que el `code` campo sea igual a «Reg».

Crea documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.patient.aggregate([
```

```
  { $redact: {
    $cond: {
      if: { $eq: ["Reg", "$code"] },
      then: "$$DESCEND",
      else: "$$PRUNE"
    }
  }}
]);
```

Salida

```
{
  "_id": 2,
  "code": "Reg",
  "patient": "Jane Doe",
  "DOB": "3/27/1989",
  "Hospital": "Cherry Hill"
}
```

Ejemplos de código

Para ver un ejemplo de código para usar el $$DESCEND comando, elija la pestaña del idioma que desee usar:

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 variable de `$$KEEP` sistema se usa con la `$redact` etapa de la canalización de agregación para mantener el documento o campo actual sin cambios e incluirlo en la salida.

Parámetros

Ninguno

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra el uso de $$KEEP en una canalización de agregación de Amazon DocumentDB. Los documentos solo se conservan si el acceso es igual a «público»; de lo contrario, se eliminan.

Cree documentos de muestra

```
db.articles.insertMany([
  { title: "Article A", access: "public", content: "Visible content" },
  { title: "Article B", access: "private", content: "Hidden content" }
]);
```

Ejemplo de consulta

```
db.articles.aggregate([
  {
    $redact: {
      $cond: [
        { $eq: ["$access", "public"] },
        "$$KEEP",
        "$$PRUNE"
      ]
    }
  }
]);
```

Salida

```
[
  {
    "_id" : ObjectId("..."),
    "title" : "Article A",
    "access" : "public",
    "content" : "Visible content"
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el $$KEEP comando, elija la pestaña del idioma que desee usar:

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 variable de $$PRUNE sistema se usa con la `$redact` etapa de la canalización de agregación para excluir los documentos o los niveles de documentos incrustados de los resultados. Cuando una condición se evalúa como$$PRUNE, el documento o subdocumento actual se elimina de la salida. Por lo general, se usa con $$DESCEND (para conservar y recorrer el documento) o $$KEEP (para mantener el documento en todos los niveles).

Parámetros

Ninguna. La variable de $$PRUNE sistema se utiliza sin ningún parámetro y se debe utilizar con`$redact`.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra cómo utilizar $$PRUNE with `$redact` para excluir de los resultados a los usuarios mayores de 30 años.

Cree documentos de muestra

```
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" } }
])
```

Ejemplo de consulta

```
db.users.aggregate([
  {
    $redact: {
      $cond: {
        if: { $gt: ["$age", 30] },
        then: "$$PRUNE",
        else: "$$DESCEND"
      }
    }
  }
])
```

Salida

```
[
  {
    "_id": 2,
    "name": "Saanvi Sarkar",
    "age": 28,
    "address": {
      "street": "456 Oak Rd",
      "city": "Someplace",
      "state": "NY"
    }
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el $$PRUNE comando, elija la pestaña del idioma que desee usar:

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()
```

$ RAÍZ

El $ROOT operador de Amazon DocumentDB se utiliza para hacer referencia a todo el documento de entrada dentro de una canalización de agregación. Le permite acceder a todo el documento y manipularlo, incluidos todos sus campos y estructuras anidados.

Parámetros

Ninguno

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo $ROOT crear un registro de auditoría que capture el documento original completo junto con los metadatos sobre el momento en que se procesó.

Cree documentos de muestra

```
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
  }
]);
```

Ejemplo de consulta

```
db.orders.aggregate([
  {
    $project: {
      processedAt: new Date(),
      originalDocument: "$$ROOT",
      summary: {
        $concat: [
          "Order ",
          "$orderId",
          " for ",
          "$customer",
          " - Total: $",
          { $toString: "$totalAmount" }
        ]
      }
    }
  }
]);
```

Salida

```
[
  {
    _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'
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el $R00T comando, elija la pestaña del idioma que desee usar:

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

El `$abs` operador de Amazon DocumentDB devuelve el valor absoluto de un número. Se puede usar en la canalización de agregación para realizar operaciones matemáticas en campos numéricos.

Parámetros

- `number`: La expresión numérica para la que se devolverá el valor absoluto.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra el uso del `$abs` operador para encontrar el valor absoluto de un campo numérico.

Cree documentos de muestra

```
db.numbers.insertMany([
  { "_id": 1, "value": -5 },
  { "_id": 2, "value": 10 },
  { "_id": 3, "value": -3.14 },
  { "_id": 4, "value": 0 }
]);
```

Ejemplo de consulta

```
db.numbers.aggregate([
```

```
  { $project: {
    "_id": 1,
    "absolute_value": { $abs: "$value" }
  }}
]);
```

Salida

```
[
  { "_id": 1, "absolute_value": 5 },
  { "_id": 2, "absolute_value": 10 },
  { "_id": 3, "absolute_value": 3.14 },
  { "_id": 4, "absolute_value": 0 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$abs` comando, elija la pestaña del idioma que desee usar:

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

Nuevo a partir de la versión 8.0.1.

El `$acos` operador de Amazon DocumentDB devuelve el arcoseno (coseno inverso) de un valor. El valor de entrada debe estar comprendido entre -1 y 1.

Parámetros

- `expression`: expresión que se resuelve en un número entre -1 y 1.

El resultado se expresa en radianes. Para obtener grados, aplíquelos `$radiansToDegrees` a la salida.

El tipo de devolución es `double` el predeterminado. Si la entrada es un decimal de 128 bits, la salida también es un decimal de 128 bits.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$acos` operador para calcular el arcoseno de los valores numéricos.

Cree documentos de muestra

```
db.values.insertMany([
  { "_id": 1, "value": 1 },
  { "_id": 2, "value": 0 },
  { "_id": 3, "value": -1 }
]);
```

Ejemplo de consulta

```
db.values.aggregate([
  { $project: {
    "angle": { $acos: "$value" }
  }}
]);
```

Salida

```
[
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 1.5707963267948966 },
  { "_id": 3, "angle": 3.141592653589793 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$acos` operador, elige la pestaña del idioma que quieres usar:

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()
```

$ un cosh

Nuevo a partir de la versión 8.0.1.

El `$acosh` operador de Amazon DocumentDB devuelve el coseno hiperbólico inverso (arcoseno hiperbólico) de un valor. El valor de entrada debe ser mayor o igual a 1.

Parámetros

- `expression`: expresión que se resuelve en un número mayor o igual a 1.

El tipo de devolución es `double` el predeterminado. Si la entrada es un decimal de 128 bits, la salida también es un decimal de 128 bits.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$acosh` operador para calcular el coseno hiperbólico inverso de los valores numéricos.

Cree documentos de muestra

```
db.values.insertMany([
  { "_id": 1, "value": 1 },
  { "_id": 2, "value": 2 },
  { "_id": 3, "value": 10 }
]);
```

Ejemplo de consulta

```
db.values.aggregate([
  { $project: {
    "result": { $acosh: "$value" }
  }}
]);
```

Salida

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 1.3169578969248166 },
  { "_id": 3, "result": 2.993222846126381 }
```

```
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$acosh` operador, elige la pestaña del idioma que quieres usar:

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

El `$add` operador de Amazon DocumentDB se utiliza para sumar números o fechas. Se puede utilizar para realizar operaciones aritméticas en campos numéricos o para realizar operaciones aritméticas de fechas añadiendo varias unidades de tiempo a un campo de fecha.

Parámetros

- `expression1`: el primer número o fecha que se va a añadir.
- `expression2`: El segundo número o fecha que se va a añadir.
- `expression3`: (opcional) Números o fechas adicionales para añadir.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$add` operador para sumar dos números.

Cree documentos de muestra

```
db.numbers.insertMany([
  { _id: 1, a: 5, b: 3 },
  { _id: 2, a: 10, b: 7 },
  { _id: 3, a: 2, b: 8 }
]);
```

Ejemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    sum: { $add: ["$a", "$b"] }
```

```
  }}
])
```

Salida

```
[
  { "_id" : 1, "sum" : 8 },
  { "_id" : 2, "sum" : 17 },
  { "_id" : 3, "sum" : 10 }
]
```

En este ejemplo, el `$add` operador se utiliza para añadir los valores de los b campos a y de cada documento y almacenar el resultado en el `sum` campo.

Ejemplos de código

Para ver un ejemplo de código para usar el `$add` comando, elija la pestaña correspondiente al idioma que desee usar:

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()
```

$añadir ToSet

El operador de `$addToSet` agregación devuelve una matriz de valores únicos de una expresión especificada para cada grupo. Se usa dentro del `$group` escenario para acumular valores distintos y eliminar automáticamente los duplicados.

Parámetros

- `expression`: La expresión que se va a evaluar para cada documento del grupo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra cómo usar el `$addToSet` operador para recopilar ciudades únicas en las que se realizaron los pedidos para cada cliente.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.orders.aggregate([
  {
    $group: {
      _id: "$customer",
      cities: { $addToSet: "$city" }
    }
  }
]);
```

Salida

```
[
  { _id: 'Bob', cities: [ 'Seattle', 'Boston' ] },
  { _id: 'Alice', cities: [ 'Seattle', 'Portland' ] }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$addToSet` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve una matriz con los valores únicos de la expresión especificada para los documentos de cada ventana. Se especifica el operador en el `output` campo y, si lo desea, se definen los límites de la ventana con un `window` documento.

> **Note**
>
> Cuando se usa como operador de ventana en`$setWindowFields`, `$addToSet` está limitado a 100 MB de datos intermedios. Una operación que supera este límite devuelve un error.

Crea documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

En el ejemplo siguiente se dividen los documentos`customer`, se ordena cada partición y se devuelve el conjunto de `city` valores únicos desde el principio de la partición hasta el documento actual. `day`

```
db.visits.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$customer",
      sortBy: { day: 1 },
      output: {
        citiesSoFar: {
          $addToSet: "$city",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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" ] }
]
```

Cada documento se complementa con `citiesSoFar` el conjunto de `city` valores únicos dentro de su partición desde el principio hasta el documento actual. Al igual que con la forma acumuladora, el orden de los valores en la matriz devuelta no está garantizado.

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$addToSet` del operador, elija la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran tanto el uso del acumulador (en`$group`) como el uso del operador de ventanas (en`$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` etapa del proceso de agregación de Amazon DocumentDB le permite añadir nuevos campos calculados a los documentos. Esto puede resultar útil para añadir datos derivados o transformados a los documentos.

Parámetros

- `newField`: el nombre del nuevo campo que se va a añadir.
- `expression`: una expresión que se resuelve en el valor del nuevo campo.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra cómo `$addFields` añadir un nuevo campo `TotalInventory` que calcule el inventario total en función de los campos `Inventory.OnHand` y`Inventory.OrderQnty`.

Cree documentos de muestra

```
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
  }
]);
```

Ejemplo de consulta

```
db.example.aggregate([
  {
    $addFields: {
      TotalInventory: { $add: ["$Inventory.OnHand", "$Inventory.OrderQnty"] }
    }
  }
])
```

Salida

```
[
  {
    "_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
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$addFields` comando, elija la pestaña del idioma que desee usar:

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()
```

$todos ElementsTrue

Nuevo desde la versión 4.0

El `$allElementsTrue` operador se utiliza para comprobar si todos los elementos de una matriz tienen un valor verdadero.

Parámetros

- `expression`: expresión que da como resultado una matriz.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso de `$allElementsTrue` para comprobar si todos los elementos de una matriz son verdaderos.

Cree documentos de muestra

```
db.collection.insert([
  { "name": "John", "scores": [100, 90, 80] },
  { "name": "Jane", "scores": [80, 85, 0] },
  { "name": "Bob", "scores": [90, 95, null] }
])
```

Ejemplo de consulta

```
db.collection.find({
  "scores": { "$allElementsTrue": [{ "$gt": 0 }] }
})
```

Salida

```
[
  { "_id" : ObjectId("..."), "name" : "John", "scores" : [ 100, 90, 80 ] },
  { "_id" : ObjectId("..."), "name" : "Bob", "scores" : [ 90, 95, null ] }
]
```

En este ejemplo, la consulta comprueba si todos los elementos de la `scores` matriz son mayores que 0. `"name": "Jane"`Se excluye el documento con porque la `scores` matriz contiene un 0, que es un valor falso.

Ejemplos de código

Para ver un ejemplo de código para usar el `$allElementsTrue` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El operador de $and agregación evalúa varias expresiones y retorna true solo si todas las expresiones tienen una evaluación igual atrue. Si alguna expresión lo esfalse, retornafalse.

Parámetros

- expressions: Un conjunto de expresiones para evaluar.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra el uso del $and operador para comprobar si los productos cumplen varios criterios.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.products.aggregate([
  {
    $project: {
      name: 1,
      price: 1,
      inStock: 1,
      affordable: {
        $and: [
          { $lt: ["$price", 100] },
          { $eq: ["$inStock", true] }
        ]
      }
```

```
    }
  }
]);
```

Salida

```
[
  { _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 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el operador de $and agregación, elija la pestaña correspondiente al idioma que desee usar:

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()
```

$cualquier ElementTrue

Nuevo desde la versión 4.0

El `$anyElementTrue` operador se usa para determinar si algún elemento de una matriz es verdadero.

Parámetros

- `field`: un campo de matriz para evaluar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso de `$anyElementTrue` para comprobar si algún elemento de una matriz es verdadero.

Cree documentos de muestra

```
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] }
])
```

Ejemplo de consulta

```
db.grades.aggregate([
  {
    $project: {
      student: 1,
      isAnyTrue: { $anyElementTrue: ["$scores"] },
      _id: 0
    }
  }
])
```

Salida

```
[
  { student: 'Tim', isAnyTrue: false },
  { student: 'Bob', isAnyTrue: false },
  { student: 'Ivy', isAnyTrue: true }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$anyElementTrue` comando, elija la pestaña del idioma que desee usar:

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()
```

$matriz ElemAt

El `$arrayElemAt` operador de Amazon DocumentDB le permite recuperar un elemento de una matriz por su posición en el índice. Esto resulta especialmente útil cuando necesita acceder a un elemento específico dentro de un campo matricial de sus documentos.

Parámetros

- `array`: La matriz de entrada de la que se va a recuperar un elemento.
- `index`: la posición de índice basada en cero del elemento que se va a recuperar. Este valor debe ser un número entero no negativo.

Ejemplo (MongoDB Shell)

En este ejemplo, demostraremos cómo usar el `$arrayElemAt` operador para recuperar elementos específicos de la `flight_miles` matriz de la `miles` colección.

Cree documentos de muestra

```
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 ]}
]);
```

Ejemplo de consulta

```
db.miles.aggregate([
  { $project: {
    "_id": 1,
    "first_mile": { $arrayElemAt: [ "$flight_miles", 0 ] },
    "last_mile": { $arrayElemAt: [ "$flight_miles", -1 ] }
  }}
]);
```

Salida

```
{ "_id" : 1, "first_mile" : 1205, "last_mile" : 880 }
{ "_id" : 2, "first_mile" : 1205, "last_mile" : 2780 }
{ "_id" : 3, "first_mile" : 1205, "last_mile" : 880 }
```

En este ejemplo, utilizamos el `$arrayElemAt` operador para recuperar el primer y el último elemento de la `flight_miles` matriz de cada documento.

Ejemplos de código

Para ver un ejemplo de código para usar el `$arrayElemAt` comando, elija la pestaña correspondiente al idioma que desee usar:

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()
```

$matriz ToObject

El `$arrayToObject` operador de Amazon DocumentDB es el inverso del `$objectToArray` operador. Toma una matriz de documentos de pares clave-valor y la convierte en un único documento. Esto resulta especialmente útil cuando se necesita volver a convertir una matriz de pares clave-valor en una estructura de objeto o documento.

Parámetros

- `array expression`: expresión que se resuelve en una matriz. Los elementos de la matriz deben ser documentos con dos campos: k (la clave) y v (el valor).

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra cómo `$arrayToObject` convertir una matriz de pares clave-valor en un documento.

Cree documentos de muestra

```
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 } }
]);
```

Ejemplo de consulta

```
db.videos.aggregate([
  { $project: { name: 1, videos: { $objectToArray: "$inventory" } } },
  { $project: { name: 1, inventory: { $arrayToObject: "$videos" } } }
]);
```

Salida

```
{ "_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 } }
```

En este ejemplo, el `$objectToArray` operador se utiliza para convertir el `inventory` objeto en una matriz de pares clave-valor. A continuación, el `$arrayToObject` operador se utiliza para volver a convertir la matriz en un documento, restaurando la estructura original del objeto.

Ejemplos de código

Para ver un ejemplo de código para usar el `$arrayToObject` comando, elija la pestaña correspondiente al idioma que desee usar:

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

Nuevo a partir de la versión 8.0.1.

El `$asin` operador de Amazon DocumentDB devuelve el arcoseno (seno inverso) de un valor. El valor de entrada debe estar comprendido entre -1 y 1.

Parámetros

- `expression`: expresión que se resuelve en un número entre -1 y 1.

El resultado se expresa en radianes. Para obtener grados, aplíquelos `$radiansToDegrees` a la salida.

El tipo de devolución es `double` el predeterminado. Si la entrada es un decimal de 128 bits, la salida también es un decimal de 128 bits.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$asin` operador para calcular el arcoseno de los valores numéricos.

Cree documentos de muestra

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 0.5 },
  { "_id": 3, "value": 1 }
]);
```

Ejemplo de consulta

```
db.values.aggregate([
  { $project: {
    "angle": { $asin: "$value" }
  }}
]);
```

Salida

```
[
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.5235987755982989 },
  { "_id": 3, "angle": 1.5707963267948966 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$asin` operador, elige la pestaña del idioma que quieres usar:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('values');

  const result = await collection.aggregate([
    { $project: {
      "angle": { $asin: "$value" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['values']

    result = list(collection.aggregate([
        { '$project': {
            'angle': { '$asin': '$value' }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$asin

Nuevo a partir de la versión 8.0.1.

El `$asinh` operador de Amazon DocumentDB devuelve el seno hiperbólico inverso de un valor. Utilícelo en el proceso de agregación para realizar cálculos trigonométricos hiperbólicos en campos numéricos.

Parámetros

- `expression`: expresión que se resuelve en un número.

El resultado está en radianes. Para obtener grados, aplíquelos `$radiansToDegrees` a la salida.

El tipo de devolución es `double` el predeterminado. Si la entrada es un decimal de 128 bits, la salida también es un decimal de 128 bits.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$asinh` operador para calcular el seno hiperbólico inverso de los valores numéricos.

Cree documentos de muestra

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Ejemplo de consulta

```
db.values.aggregate([
  { $project: {
    "result": { $asinh: "$value" }
  }}
]);
```

Salida

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 0.881373587019543 },
  { "_id": 3, "result": -0.881373587019543 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$asinh` operador, elige la pestaña del idioma que quieres usar:

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()
```

$ atán

Nuevo a partir de la versión 8.0.1.

El `$atan` operador de Amazon DocumentDB devuelve la tangente inversa (arcotangente) de un valor.

La expresión de entrada debe convertirse en un valor numérico.

El resultado se expresa en radianes. Para obtener grados, aplíquelos `$radiansToDegrees` a la salida.

El tipo de devolución es `double` el predeterminado. Si la entrada es un decimal de 128 bits, la salida también es un decimal de 128 bits.

Parámetros

- `expression`: expresión que se resuelve en un número.

Comportamiento

null y NaN

Ejemplo	Resultados
`{ $atan: NaN }`	`NaN`
`{ $atan: null }`	`null`

Cuando la entrada es `null` o falta el campo al que se hace referencia, el resultado es`null`. Una entrada de `NaN` productos`NaN`.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$atan` operador para calcular el arcotangente de un valor.

Cree documentos de muestra

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
```

```
]);
```

Ejemplo de consulta

```
db.values.aggregate([
  { $project: {
    "angle": { $atan: "$value" }
  }}
]);
```

Salida

```
[
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.7853981633974483 },
  { "_id": 3, "angle": -0.7853981633974483 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$atan` operador, elige la pestaña del idioma que quieres usar:

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

Nuevo a partir de la versión 8.0.1.

El `$atan2` operador de Amazon DocumentDB devuelve la tangente inversa (arcotangente) dey `/ x`, donde y y `x` son los valores primero y segundo pasados a la expresión, respectivamente.

Ambas expresiones deben convertirse en valores numéricos. El operador usa el signo de ambos argumentos para determinar el cuadrante correcto.

El resultado está en radianes. Para obtener grados, aplíquelos `$radiansToDegrees` a la salida.

El tipo de devolución es `double` el predeterminado. Si las entradas son decimales de 128 bits, la salida también es un decimal de 128 bits.

Parámetros

- `expression 1`(y): expresión que se resuelve en un número que representa la coordenada y.
- `expression 2`(x): expresión que se resuelve en un número que representa la coordenada x.

Comportamiento

null y NaN

Si alguno de los argumentos es `null` (o hace referencia a un campo que falta), el resultado es`null`. Si alguno de los argumentos `NaN` lo es, el resultado es`NaN`. Cuando un argumento es `null` y el otro es`NaN`, `null` tiene prioridad.

Ejemplo	Resultados
`{ $atan2: [ NaN, <value> ] }`	`NaN`
`{ $atan2: [ <value>, NaN ] }`	`NaN`
`{ $atan2: [ null, <value> ] }`	`null`
`{ $atan2: [ <value>, null ] }`	`null`
`{ $atan2: [ NaN, null ] }`	`null`
`{ $atan2: [ null, NaN ] }`	`null`

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$atan2` operador para calcular el arcotangente de un conjunto y/x de puntos.

Cree documentos de muestra

```
db.points.insertMany([
  { "_id": 1, "y": 1, "x": 1 },
  { "_id": 2, "y": 1, "x": 0 },
  { "_id": 3, "y": -1, "x": -1 }
]);
```

Ejemplo de consulta

```
db.points.aggregate([
  { $project: {
    "angle": { $atan2: ["$y", "$x"] }
  }}
]);
```

Salida

```
[
  { "_id": 1, "angle": 0.7853981633974483 },
  { "_id": 2, "angle": 1.5707963267948966 },
  { "_id": 3, "angle": -2.356194490192345 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$atan2` operador, elige la pestaña del idioma que quieres usar:

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

Nuevo a partir de la versión 8.0.1.

El `$atanh` operador de Amazon DocumentDB devuelve la tangente hiperbólica inversa (arcotangente hiperbólica) de un valor.

La expresión de entrada debe tener como resultado un número comprendido en el rango de (inclusive). `-1` 1 Los valores que están fuera de este rango provocan un error.

El tipo de devolución es `double` el predeterminado. Si la entrada es un decimal de 128 bits, la salida también es un decimal de 128 bits.

Parámetros

- `expression`: expresión que se resuelve en un número comprendido entre `-1` y, ambos inclusive. 1

Comportamiento

null, NaN y +/- Infinity

Ejemplo	Resultados
`{ $atanh: NaN }`	`NaN`
`{ $atanh: null }`	`null`
`{ $atanh: 1 }`	`Infinity`
`{ $atanh: -1 }`	`-Infinity`
`{ $atanh: Infinity }` o `{ $atanh: -Infinity }`	Lanza un error.

Cuando la entrada es `null` o falta el campo al que se hace referencia, el resultado es`null`. Una entrada de `NaN` productos`NaN`. Los valores límite `1` y `-1` produce `Infinity` y`-Infinity`, respectivamente. Cualquier valor fuera del `[-1, 1]` rango (incluido`±Infinity`) provoca un error.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$atanh` operador para calcular la tangente hiperbólica inversa de un valor.

Cree documentos de muestra

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 0.5 },
  { "_id": 3, "value": -0.5 }
]);
```

Ejemplo de consulta

```
db.values.aggregate([
  { $project: {
    "result": { $atanh: "$value" }
  }}
```

```
]);
```

Salida

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 0.5493061443340548 },
  { "_id": 3, "result": -0.5493061443340548 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$atanh` operador, elige la pestaña del idioma que quieres usar:

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

El operador de `$avg` agregación de Amazon DocumentDB calcula el valor promedio de la expresión especificada en todos los documentos que se introducen en el escenario. Este operador es útil para calcular el promedio de un campo numérico o una expresión en un conjunto de documentos.

Parámetros

- `expression`: La expresión que se va a utilizar para calcular el promedio. Puede ser una ruta de campo (por ejemplo`"$field"`) o una expresión (por ejemplo`{ $multiply: ["$field1", "$field2"] }`).

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$avg` operador para calcular la puntuación media de un conjunto de documentos de los alumnos.

Cree documentos de muestra

```
db.students.insertMany([
  { name: "John", score: 85 },
  { name: "Jane", score: 92 },
```

```
  { name: "Bob", score: 78 },
  { name: "Alice", score: 90 }
]);
```

Ejemplo de consulta

```
db.students.aggregate([
  { $group: {
    _id: null,
    avgScore: { $avg: "$score" }
  }}
]);
```

Salida

```
[
  {
    "_id": null,
    "avgScore": 86.25
  }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$avg` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve el promedio de la expresión especificada para los documentos de cada ventana. Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

En el ejemplo siguiente se dividen los documentos`city`, se ordena cada partición y se devuelve una media acumulada `day` `temp` desde el principio de la partición hasta el documento actual.

```
db.readings.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$city",
      sortBy: { day: 1 },
      output: {
        runningAvg: {
          $avg: "$temp",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Cada documento se complementa con `runningAvg` el promedio `temp` de su partición hasta el documento actual, incluido.

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$avg` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran tanto el uso del acumulador (en`$group`) como el uso del operador de ventanas (en`$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()
```

$binarySize

Nuevo desde la versión 8.0.1.

El `$binarySize` operador de Amazon DocumentDB devuelve el tamaño en bytes de una cadena o un valor de datos binarios determinado. En el caso de los valores de cadena, se trata del recuento de bytes UTF-8 codificados, no del recuento de caracteres. Multi-byte UTF-8 los caracteres (como los caracteres acentuados o los caracteres CJK) producirán un recuento de bytes mayor que el número de caracteres de la cadena.

Parámetros

- `expression`: expresión que se resuelve en una cadena o un valor de datos binarios.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$binarySize` operador para devolver el tamaño en bytes de los campos de cadena.

Cree documentos de muestra

```
db.docs.insertMany([
```

```
  {_id: 1, content: "hello"},
  {_id: 2, content: "Amazon DocumentDB"},
  {_id: 3, content: ""}
]);
```

Ejemplo de consulta

```
db.docs.aggregate([
  { $project: { size: { $binarySize: "$content" } } }
]);
```

Salida

```
[
  {_id: 1, size: 5},
  {_id: 2, size: 17},
  {_id: 3, size: 0}
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$binarySize` operador, elige la pestaña del idioma que quieres usar:

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 y

Nuevo a partir de la versión 8.0.1.

El `$bitAnd` operador de Amazon DocumentDB realiza una operación AND bit a bit en valores enteros o largos.

Parámetros

- `expressions`: una matriz de dos o más expresiones, que se puede resolver en números enteros o largos.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$bitAnd` operador para ejecutar AND bit a bit en dos campos.

Cree documentos de muestra

```
db.flags.insertMany([
  {_id: 1, a: 13, b: 10},
  {_id: 2, a: 7, b: 5},
  {_id: 3, a: 15, b: 9}
]);
```

Ejemplo de consulta

```
db.flags.aggregate([
  { $project: { result: { $bitAnd: ["$a", "$b"] } } }
]);
```

Salida

```
[
  {_id: 1, result: 8},
  {_id: 2, result: 5},
  {_id: 3, result: 9}
]
```

En binario: 13 (1101) Y 10 (1010) = 8 (1000); 7 (0111) Y 5 (0101) = 5 (0101); 15 (1111) Y 9 (1001) = 9 (1001).

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$bitAnd` operador, elija la pestaña correspondiente al idioma que desee utilizar:

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()
```

$bitNo

Nuevo a partir de la versión 8.0.1.

El `$bitNot` operador de Amazon DocumentDB realiza una operación NOT bit a bit sobre un valor entero o largo y devuelve el complemento bit a bit. Para un entero determinado o largo n, el resultado es - (n+1).

Parámetros

- `expression`: una expresión que se resuelve en un entero o un largo. Devuelve el complemento bit a bit, que es - (n+1) para un entero determinado o un n largo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$bitNot` operador para calcular el complemento bit a bit de los valores enteros.

Cree documentos de muestra

```
db.numbers.insertMany([
  {_id: 1, value: 0},
  {_id: 2, value: 5},
  {_id: 3, value: -3}
]);
```

Ejemplo de consulta

```
db.numbers.aggregate([
  { $project: { result: { $bitNot: "$value" } } }
]);
```

Salida

```
[
  {_id: 1, result: -1},
  {_id: 2, result: -6},
  {_id: 3, result: 2}
]
```

En binario: NO 0 = -1; NO 5 = -6; NO -3 = 2.

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$bitNot` del operador, elige la pestaña correspondiente al idioma que deseas usar:

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

Nuevo a partir de la versión 8.0.1.

El `$bitOr` operador de Amazon DocumentDB realiza una operación OR bit a bit en valores enteros o largos.

Parámetros

- `expressions`: una matriz de dos o más expresiones, que se puede resolver en números enteros o largos.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$bitOr` operador para realizar operaciones OR a nivel de bits en dos campos.

Cree documentos de muestra

```
db.flags.insertMany([
  {_id: 1, a: 13, b: 10},
  {_id: 2, a: 7, b: 5},
  {_id: 3, a: 15, b: 9}
]);
```

Ejemplo de consulta

```
db.flags.aggregate([
  { $project: { result: { $bitOr: ["$a", "$b"] } } }
]);
```

Salida

```
[
  {_id: 1, result: 15},
  {_id: 2, result: 7},
  {_id: 3, result: 15}
]
```

En binario: 13 (1101) OR 10 (1010) = 15 (1111); 7 (0111) OR 5 (0101) = 7 (0111); 15 (1111) O 9 (1001) = 15 (1111).

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$bitOr` operador, elige la pestaña correspondiente al idioma que deseas usar:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('flags');
    const result = await collection.aggregate([
      { $project: { result: { $bitOr: ["$a", "$b"] } } }
    ]).toArray();
    console.log(result);
  } finally {
    await client.close();
  }
}
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    try:
        db = client['test']
        collection = db['flags']
        result = list(collection.aggregate([
            {'$project': {'result': {'$bitOr': ['$a', '$b']}}}
        ]))
        print(result)
    finally:
        client.close()

example()
```

$ bitX o

Nuevo desde la versión 8.0.1.

El `$bitXor` operador de Amazon DocumentDB realiza una operación XOR bit a bit en valores enteros o largos.

Parámetros

- `expressions`: una matriz de dos o más expresiones, que se puede resolver en números enteros o largos.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$bitXor` operador para realizar un XOR bit a bit en dos campos.

Cree documentos de muestra

```
db.flags.insertMany([
  {_id: 1, a: 13, b: 10},
  {_id: 2, a: 7, b: 5},
  {_id: 3, a: 15, b: 9}
]);
```

Ejemplo de consulta

```
db.flags.aggregate([
  { $project: { result: { $bitXor: ["$a", "$b"] } } }
]);
```

Salida

```
[
  {_id: 1, result: 7},
  {_id: 2, result: 2},
  {_id: 3, result: 6}
]
```

En binario: 13 (1101) XOR 10 (1010) = 7 (0111); 7 (0111) XOR 5 (0101) = 2 (0010); 15 (1111) XOR 9 (1001) = 6 (0110).

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$bitXor` operador, elija la pestaña correspondiente al idioma que desee utilizar:

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()
```

$inferior

Nuevo a partir de la versión 8.0.1.

Utilice el `$bottom` acumulador de la `$group` fase para obtener el documento con la clasificación más baja por grupo según un orden de clasificación especificado.

Parámetros

- `sortBy`: un documento que especifica el orden de clasificación. 1Utilícelo para ascender o `-1` descender.
- `output`: expresión que especifica los campos que se devolverán desde la parte inferior del documento.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$bottom` acumulador para encontrar la venta más baja (la cantidad más baja) por artículo de una colección de ventas.

Crea documentos de muestra

```
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 }
])
```

Ejemplo de consulta

```
db.sales.aggregate([
  { $group: { _id: "$item", bottomSale: { $bottom: { sortBy: { quantity: -1 }, output:
 { quantity: "$quantity", price: "$price" } } } } }
])
```

Salida

```
[
  { "_id": "xyz", "bottomSale": { "quantity": 7, "price": 6 } },
  { "_id": "abc", "bottomSale": { "quantity": 5, "price": 8 } }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$bottom` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve el `output` documento que ocupa el último lugar (según el documento del operador`sortBy`) de entre los documentos de cada ventana. Se especifica el operador en el `output` campo y, si lo desea, se definen los límites de la ventana con un `window` documento.

> **Note**
>
> Cuando se usa como operador de ventana en`$setWindowFields`, `$bottom` está limitado a 100 MB de datos intermedios. Una operación que supera este límite devuelve un error.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

El ejemplo siguiente divide los documentos por`player`, ordena cada partición y devuelve la ronda con la puntuación más baja vista desde el principio de la partición hasta el documento actual. `round`

```
db.matchScores.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$player",
      sortBy: { round: 1 },
```

```
      output: {
        worstRoundSoFar: {
          $bottom: { sortBy: { score: -1 }, output: { round: "$round", score:
 "$score" } },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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 } }
]
```

Cada documento se complementa con `worstRoundSoFar` la ronda con la puntuación más baja de su partición hasta el documento actual, incluido.

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$bottom` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran tanto el uso del acumulador (en`$group`) como el uso del operador de ventanas (en`$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()
```

$ a N

Nuevo a partir de la versión 8.0.1.

Usa el `$bottomN` acumulador del `$group` escenario para devolver los N elementos inferiores de un grupo según un orden de clasificación especificado. Si un grupo contiene menos de N elementos, `$bottomN` devuelve todos los elementos del grupo.

Parámetros

- n: un entero positivo o una expresión que se resuelve en uno y que especifica cuántos resultados inferiores se devolverán por grupo.
- `sortBy`: un documento que especifica el orden de clasificación. 1Utilícelo para ascender o `-1` descender.
- `output`: expresión que especifica los campos que se devolverán de cada uno de los N documentos de la parte inferior.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$bottomN` acumulador para encontrar las dos últimas ventas (la cantidad más baja) por artículo de una colección de ventas.

Crea documentos de muestra

```
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 }
])
```

Ejemplo de consulta

```
db.sales.aggregate([
  { $group: { _id: "$item", bottomTwoSales: { $bottomN: { n: 2, sortBy: { quantity:
 -1 }, output: { quantity: "$quantity", price: "$price" } } } } }
])
```

Salida

```
[
  { "_id": "xyz", "bottomTwoSales": [{ "quantity": 9, "price": 6 }, { "quantity": 3,
 "price": 12 }] },
  { "_id": "abc", "bottomTwoSales": [{ "quantity": 7, "price": 8 }, { "quantity": 5,
 "price": 10 }] }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$bottomN` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve una matriz de los N documentos `output` de la parte inferior (según los propios documentos del operador`sortBy`) de entre los documentos de cada ventana. Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

> **Note**
>
> Cuando se usa como operador de ventana en`$setWindowFields`, `$bottomN` está limitado a 100 MB de datos intermedios. Una operación que supera este límite devuelve un error.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

El ejemplo siguiente divide los documentos`player`, ordena cada partición y devuelve las dos rondas con la puntuación más baja desde el principio de la partición hasta el documento actual. `round`

```
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"] }
        }
```

```
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }] }
]
```

Cada documento se complementa con `bottomTwoRoundsSoFar` las dos rondas con la puntuación más baja de su partición hasta el documento actual, incluido. Si la ventana contiene menos de dos documentos, se devuelven todos.

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$bottomN` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran tanto el uso del acumulador (en`$group`) como el uso del operador de ventanas (en`$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

Nuevo desde la versión 8.0.1.

El `$bsonSize` operador de Amazon DocumentDB devuelve el tamaño en bytes de un documento codificado como BSON.

Parámetros

- `expression`: una expresión que se convierte en un documento u objeto.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$bsonSize` operador para devolver el tamaño BSON de cada documento.

Cree documentos de muestra

```
db.items.insertMany([
  {_id: 1, name: "a", tags: ["x"]},
  {_id: 2, name: "ab", tags: ["x", "y", "z"]},
  {_id: 3, name: "abc", tags: []}
]);
```

Ejemplo de consulta

```
db.items.aggregate([
  { $project: { docSize: { $bsonSize: "$$ROOT" } } }
]);
```

Salida

```
[
  {_id: 1, docSize: 46},
  {_id: 2, docSize: 65},
  {_id: 3, docSize: 39}
]
```

> **Note**
>
> Los tamaños exactos dependen de la codificación BSON de los campos del documento.

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$bsonSize` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar:

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

Novedades de la versión 8.0

No es compatible con el clúster de Elastic.

La etapa de `$bucket` agregación de Amazon DocumentDB le permite agrupar los documentos de entrada en cubos en función de una expresión específica y de los límites de los cubos. Esto puede resultar útil para analizar los datos que se encuentran dentro de determinados rangos de valores o categorías.

Parámetros

- `groupBy`(obligatorio): la expresión que especifica el valor por el que se va a agrupar.
- `boundaries`(obligatorio): matriz de valores dobles que definen los límites de los cubos. Los documentos se asignan a los cubos en función del valor de `groupBy` expresión que se encuentra dentro de los límites especificados.
- `default`(opcional): un valor literal que se genera para los documentos cuyo valor de `groupBy` expresión no se encuentra dentro de ninguno de los límites especificados.
- `output`(opcional): objeto que especifica la información que se va a generar para cada segmento. Puede usar operadores acumuladores como`$sum`, `$avg``$min`, y `$max` para calcular las agregaciones de cada segmento.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$bucket` escenario para agrupar los datos de ventas por rango de precios.

Cree documentos de muestra

```
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") }
]);
```

Ejemplo de consulta

```
db.sales.aggregate([
  {
    $bucket: {
```

```
      groupBy: "$price",
      boundaries: [0, 10, 20, 30],
      default: "Other",
      output: {
        "count": { $sum: 1 },
        "totalQuantity": { $sum: "$quantity" }
      }
    }
  },
  {
    $sort: { _id: 1 }
  }
])
```

Salida

```
[
  { _id: 0, count: 1, totalQuantity: 3 },
  { _id: 10, count: 2, totalQuantity: 4 },
  { _id: 20, count: 2, totalQuantity: 2 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$bucket` comando, elija la pestaña del idioma que desee usar:

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

Nuevo a partir de la versión 8.0.2.

La fase de `$bucketAuto` agregación de Amazon DocumentDB agrupa automáticamente los documentos de entrada en un número específico de cubos en función de una expresión específica. A diferencia `$bucket` de lo que ocurre cuando se definen los límites de los cubos de forma explícita, los determina `$bucketAuto` automáticamente para que los documentos se distribuyan de la forma más uniforme posible en el número de cubos solicitado. Esto puede resultar útil cuando desea clasificar los datos en un número fijo de grupos sin conocer de antemano la distribución de los valores.

Parámetros

- `groupBy`(obligatorio): expresión que especifica el valor por el que se va a agrupar.
- `buckets`(obligatorio): un entero positivo que especifica el número de cubos en los que se agrupan los documentos. En función de los documentos de entrada y de la `groupBy` expresión, el número real de cubos puede ser inferior al número especificado.
- `output`(opcional): objeto que especifica la información que se generará para cada bucket. Puedes usar operadores acumuladores como`$sum`, `$avg``$min`, y `$max` para calcular las agregaciones de cada depósito. Si se omite, cada bucket genera un `count` campo de forma predeterminada.
- `granularity`(opcional): cadena que especifica la serie numérica que se prefiere usar para los límites del bucket. Esto ajusta los límites calculados a una progresión numérica estándar. Este parámetro solo está disponible si todos los `groupBy` valores son numéricos y ninguno de ellos lo es. `NaN` Los valores válidos son: `R5``R10`,`R20`,`R40`,`R80`,`1-2-5`,`E6`,`E12`,`E24`,`E48`, `E96``E192`, y`POWERSOF2`.

Salida

Cada bucket se devuelve como un documento independiente que contiene un `_id` objeto que describe los límites del bucket:

- `_id.min`: El límite inferior inclusivo del bucket.
- `_id.max`: El límite superior del cubo. Este límite es exclusivo para todos los cubos, excepto para el último segmento de la serie, en el que es inclusivo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$bucketAuto` escenario para agrupar los datos de ventas en grupos por precio.

Cree documentos de muestra

```
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") }
]);
```

Ejemplo de consulta

El siguiente comando agrupa los documentos de muestra en tres grupos por precio:

```
db.sales.aggregate([
  {
    $bucketAuto: {
      groupBy: "$price",
      buckets: 3,
      output: {
        "count": { $sum: 1 },
        "totalQuantity": { $sum: "$quantity" }
      }
    }
  }
])
```

Salida

El comando devuelve el siguiente resultado:

```
[
  { _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 }
]
```

Ejemplo de consulta con granularidad

El siguiente ejemplo agrega la `granularity` opción de ajustar los límites calculados del bucket a potencias de 2.

```
db.sales.aggregate([
  {
    $bucketAuto: {
      groupBy: "$price",
      buckets: 2,
      granularity: "POWERSOF2",
      output: {
        "count": { $sum: 1 },
        "totalQuantity": { $sum: "$quantity" }
      }
    }
  }
])
```

Salida

El comando devuelve el siguiente resultado:

```
[
  { _id: { min: 4, max: 16 }, count: 3, totalQuantity: 7 },
  { _id: { min: 16, max: 32 }, count: 2, totalQuantity: 2 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$bucketAuto` comando, elige la pestaña del idioma que deseas usar:

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

Novedades de la versión 4.0

El `$ceil` operador de Amazon DocumentDB, al igual que en MongoDB, redondea un número al entero más cercano. Esto resulta útil cuando necesita realizar operaciones matemáticas en campos numéricos y asegurarse de que el resultado es un número entero.

Parámetros

- `expression`: la expresión numérica que se va a redondear.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo utilizar el `$ceil` operador para redondear un campo numérico.

Cree documentos de muestra

```
db.numbers.insertMany([
  { "_id": 1, "value": 3.14 },
  { "_id": 2, "value": -2.7 },
  { "_id": 3, "value": 0 }
])
```

Ejemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    "roundedUp": { $ceil: "$value" }
  }}
])
```

Salida

```
{ "_id": 1, "roundedUp": 4 }
{ "_id": 2, "roundedUp": -2 }
{ "_id": 3, "roundedUp": 0 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$ceil` comando, elija la pestaña del idioma que desee usar:

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()
```

$changeStream

Elastic Cluster no lo admite.

La etapa de `$changeStream` agregación abre un cursor de flujo de cambios para monitorear los cambios en tiempo real en una colección. Devuelve los documentos de eventos de cambio cuando se realizan operaciones de inserción, actualización, reemplazo o eliminación.

Parámetros

- `fullDocument`: especifica si se debe devolver el documento completo para las operaciones de actualización. Las opciones son `default` y `updateLookup`.
- `resumeAfter`: opcional. Reanuda el token para continuar desde un punto específico del flujo de cambios.
- `startAtOperationTime`: opcional. Marca de tiempo desde la que iniciar el flujo de cambios.
- `allChangesForCluster`: opcional. Valor booleano. Cuándo`true`, observa todos los cambios en el clúster (para la base de datos de administración). Cuando `false` (predeterminado), solo observa la colección especificada.

- `showExpandedEvents`: opcional. Valor booleano. Cuando`true`, incluye las notificaciones de eventos del lenguaje de definición de datos ampliado (DDL), como `createCollection` y`createIndexes`, en el flujo de cambios. Cuando `false` (predeterminado), estos eventos no se devuelven. Disponible en Amazon DocumentDB 8.0 a partir de la versión 8.0.2 del motor.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra cómo usar el `$changeStream` escenario para supervisar los cambios en una colección.

Ejemplo de consulta

```
// 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()));
}
```

Salida

```
{
  _id: { _data: '...' },
  operationType: 'insert',
  clusterTime: Timestamp(1, 1234567890),
  fullDocument: { _id: 1, item: 'Widget', qty: 10 },
  ns: { db: 'test', coll: 'inventory' },
  documentKey: { _id: 1 }
}
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso de la etapa de `$changeStream` agregación, elija la pestaña correspondiente al idioma que desea usar:

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()
```

$cambiar StreamSplitLargeEvent

Los clústeres elásticos no admiten esta etapa de agregación.

Esta etapa está disponible a partir de la versión 8.0.2 del motor Amazon DocumentDB.

La fase de `$changeStreamSplitLargeEvent` agregación se utiliza dentro de una canalización de $ChangeStream para dividir los eventos que superen los 16 MB en fragmentos más pequeños. Devuelve cada fragmento de forma secuencial mediante el cursor de flujo de cambios.

Parámetros

Ninguna. La entrada al `$changeStreamSplitLargeEvent` escenario debe ser un documento vacío.

Ejemplo: MongoDB Shell

El siguiente ejemplo demuestra cómo usar el `$changeStreamSplitLargeEvent` escenario para dividir un evento de cambio sobredimensionado en varios fragmentos.

Ejemplo de consulta

```
// 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()));
}
```

Salida

```
{
  _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: [] }
}
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso de la etapa de `$changeStreamSplitLargeEvent` agregación, elija la pestaña correspondiente al idioma que desea usar:

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

El `$cmp` operador de Amazon DocumentDB se utiliza para comparar dos valores y devolver un valor entero que indica su orden relativo. Es un operador de comparación que compara dos expresiones y devuelve un valor entero de -1, 0 o 1, en función de si el primer valor es menor, igual o mayor que el segundo, respectivamente.

Parámetros

- `expression1`: la primera expresión que se va a comparar.
- `expression2`: La segunda expresión que se va a comparar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del $cmp operador para comparar dos valores numéricos.

Cree documentos de muestra

```
db.collection.insertMany([
  { _id: 1, value1: 10, value2: 20 },
  { _id: 2, value1: 15, value2: 15 },
  { _id: 3, value1: 20, value2: 10 }
]);
```

Ejemplo de consulta

```
db.collection.find({
  $expr: {
    $cmp: ["$value1", "$value2"]
  }
})
```

Salida

```
[
  { "_id" : 1, "value1" : 10, "value2" : 20 },
  { "_id" : 3, "value1" : 20, "value2" : 10 }
]
```

En este ejemplo, el $cmp operador compara los value2 campos value1 y de cada documento. El resultado es:

```
- `$cmp: ["$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).
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del $cmp comando, elija la pestaña correspondiente al idioma que desee utilizar:

Node.js

Este es un ejemplo del uso del $cmp operador en una Node.js aplicación con el mongodb controlador:

```
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

Este es un ejemplo del uso del $cmp operador en una aplicación de Python con el pymongo controlador:

```
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()
```

El resultado de los ejemplos de Python Node.js y el de Python será el mismo que el del ejemplo de MongoDB Shell:

```
[
  { "_id" : 1, "value1" : 10, "value2" : 20 },
  { "_id" : 2, "value1" : 15, "value2" : 15 },
  { "_id" : 3, "value1" : 20, "value2" : 10 }
]
```

$collStats

Novedades de la versión 4.0

La etapa de `$collStats` agregación de Amazon DocumentDB proporciona estadísticas sobre la colección especificada, de forma similar al `db.collection.stats()` comando del shell de MongoDB. Esta etapa se puede utilizar para recuperar información sobre la recopilación, como la cantidad de documentos, el tamaño total de la colección y varias métricas de rendimiento.

Parámetros

- `latencyStats`: (opcional) Documento que especifica las opciones para recopilar estadísticas de latencia. Este parámetro no se admite en Amazon DocumentDB.
- `recordStats`: (opcional) Documento que especifica las opciones para recopilar estadísticas de registros. Este parámetro no se admite en Amazon DocumentDB.
- `queryExecStats`: (opcional) Documento que especifica las opciones para recopilar estadísticas de ejecución de consultas. Este parámetro no se admite en Amazon DocumentDB.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar la etapa de `$collStats` agregación para recuperar estadísticas sobre una colección nombrada `test` en la db base de datos.

Cree documentos de muestra

```
db.test.insertMany([
  { "name": "John", "age": 30 },
  { "name": "Jane", "age": 25 },
  { "name": "Bob", "age": 35 }
]);
```

Ejemplo de consulta

```
db.test.aggregate([
  { $collStats: {} }
]);
```

Salida

```
{
  "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)
}
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$collStats` comando, elija la pestaña del idioma que desee usar:

Node.js

A continuación, se muestra un ejemplo de cómo utilizar la etapa de `$collStats` agregación en una Node.js aplicación que utiliza el controlador oficial de Node.js MongoDB:

```
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

A continuación, se muestra un ejemplo de cómo utilizar la etapa de `$collStats` agregación en una aplicación de Python mediante el PyMongo controlador:

```
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

El operador de `$concat` agregación de Amazon DocumentDB concatena (o combina) varias cadenas de un documento para producir una sola cadena que se puede devolver a la aplicación. Esto reduce el trabajo realizado en la aplicación, ya que la manipulación de las cadenas se realiza a nivel de la base de datos.

Parámetros

- `expression1`: La primera cadena que se va a concatenar.

- `expression2`: La segunda cadena que se va a concatenar.
- `...`: cadenas adicionales para concatenar (opcional).

Ejemplo (MongoDB Shell)

En este ejemplo, concatenamos los nombres y apellidos de los usuarios para obtener el nombre completo de cada persona.

Cree documentos de muestra

```
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"}
])
```

Ejemplo de consulta

```
db.people.aggregate([
  { $project: { full_name: { $concat: [ "$first_name", " ", "$last_name"] } } }
])
```

Salida

```
{ "_id" : 1, "full_name" : "Jane Doe" }
{ "_id" : 2, "full_name" : "John Doe" }
{ "_id" : 3, "full_name" : "Steve Smith" }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$concat` comando, elija la pestaña del idioma que desee usar:

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

El operador de `$concatArrays` agregación de Amazon DocumentDB se utiliza para concatenar dos o más matrices en una sola matriz. Esto puede resultar útil cuando necesite combinar varias matrices de datos en una sola matriz para su posterior procesamiento o análisis.

Parámetros

- `array1`: La primera matriz que se va a concatenar.
- `array2`: La segunda matriz que se va a concatenar.
- `[array3, ...]`: (opcional) Matrices adicionales que se van a concatenar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$concatArrays` operador para combinar dos matrices en una sola matriz.

Cree documentos de muestra

```
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"]
  }
]);
```

Ejemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
      _id: 0,
      name: 1,
      all_activities: { $concatArrays: ["$hobbies", "$skills"] }
    }
  }
]);
```

Salida

```
[
  {
    "name": "John Doe",
    "all_activities": [
      "reading",
      "swimming",
      "programming",
      "design"
    ]
  },
  {
    "name": "Jane Smith",
    "all_activities": [
      "hiking",
      "cooking",
      "marketing",
      "analysis"
    ]
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$concatArrays` comando, elija la pestaña del idioma que desee usar:

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

El `$cond` operador de Amazon DocumentDB se usa para evaluar una expresión condicional y devolver una de las dos expresiones de resultado posibles.

Parámetros

- `if`: la expresión booleana que se va a evaluar.
- `then`: La expresión que se devolverá si la `if` expresión es verdadera.
- `else`: La expresión que se devolverá si la `if` expresión es falsa.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$cond` operador para devolver un valor basado en la edad de una persona.

Cree documentos de muestra

```
db.people.insertMany([
  { _id: 1, name: "John Doe", age: 35 },
  { _id: 2, name: "Jane Doe", age: 25 },
  { _id: 3, name: "Bob Smith", age: 65 }
]);
```

Ejemplo de consulta

```
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"
            }
          }
        }
      }
    }
  }
])
```

Salida

```
[
  { "_id" : 1, "name" : "John Doe", "ageGroup" : "middle-aged" },
  { "_id" : 2, "name" : "Jane Doe", "ageGroup" : "young" },
  { "_id" : 3, "name" : "Bob Smith", "ageGroup" : "elderly" }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$cond` comando, elige la pestaña del idioma que quieres usar:

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()
```

$convert

Novedades de la versión 4.0

El `$convert` operador de Amazon DocumentDB se utiliza para convertir un valor de un tipo de datos a otro. Este operador resulta útil cuando necesita realizar operaciones con datos de distintos tipos, como convertir una cadena en un número o una fecha en una marca de tiempo.

Parámetros

- `to`: el tipo de datos de destino al que se va a convertir el valor. Los valores admitidos son `"string"`, `"double"`, `"long"`, `"int"` y `"date"` y `"boolean"`.
- `from`: el tipo de datos actual del valor. Si no se especifica, Amazon DocumentDB intentará detectar automáticamente el tipo de datos.
- `onError`: (opcional) El valor que se devolverá si se produce un error en la conversión. Puede ser un valor específico o uno de los siguientes valores especiales:`"null"`,`"zerofill"`, o`"error"`.
- `onNull`: (opcional) El valor que se devolverá si el valor de entrada es`null`. Puede ser un valor específico o uno de los siguientes valores especiales:`"null"`,`"zerofill"`, o`"error"`.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra la conversión de un valor de cadena en una fecha mediante el `$convert` operador.

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.users.aggregate([
  {
    $project: {
      _id: 1,
      name: 1,
```

```
      joinedOn: {
        $convert: {
          input: "$joinedOn",
          to: "date",
          onError: "null",
          onNull: "null"
        }
      }
    }
  }
])
```

Salida

```
[
  { "_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 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$convert` comando, elija la pestaña del idioma que desee usar:

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

Nuevo a partir de la versión 8.0.1.

El `$cos` operador de Amazon DocumentDB devuelve el coseno de un valor que se mide en radianes. Utilícelo en el proceso de agregación para realizar cálculos trigonométricos en campos numéricos.

Parámetros

- `expression`: expresión que se resuelve en un número en radianes.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$cos` operador para calcular el coseno de los valores del ángulo en radianes.

Cree documentos de muestra

```
db.angles.insertMany([
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 1.0471975511965976 },
  { "_id": 3, "angle": 3.141592653589793 }
]);
```

Ejemplo de consulta

```
db.angles.aggregate([
  { $project: {
    "cosine": { $cos: "$angle" }
  }}
]);
```

Salida

```
[
  { "_id": 1, "cosine": 1 },
  { "_id": 2, "cosine": 0.5000000000000001 },
  { "_id": 3, "cosine": -1 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$cos` operador, elige la pestaña del idioma que quieres usar:

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

Nuevo a partir de la versión 8.0.1.

El `$cosh` operador de Amazon DocumentDB devuelve el coseno hiperbólico de un valor. Utilícelo en el proceso de agregación para realizar cálculos trigonométricos hiperbólicos en campos numéricos.

Parámetros

- `expression`: expresión que se resuelve en un número.

El tipo de devolución es `double` el predeterminado. Si la entrada es un decimal de 128 bits, la salida también es un decimal de 128 bits.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$cosh` operador para calcular el coseno hiperbólico de los valores numéricos.

Cree documentos de muestra

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Ejemplo de consulta

```
db.values.aggregate([
  { $project: {
    "result": { $cosh: "$value" }
  }}
]);
```

Salida

```
[
  { "_id": 1, "result": 1 },
  { "_id": 2, "result": 1.5430806348152437 },
  { "_id": 3, "result": 1.5430806348152437 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$cosh` operador, elige la pestaña del idioma que quieres usar:

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 (etapa)

La fase de `$count` agregación de Amazon DocumentDB se utiliza para contar el número de documentos que pasan a la fase. Suele utilizarse como la etapa final de un proceso de agregación para obtener el recuento total de documentos que coinciden con las etapas anteriores.

Parámetros

- `field`: El campo que se va a contar. Este parámetro es opcional y, si no se proporciona, la etapa contará el número total de documentos de entrada.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$count` escenario para obtener el número total de documentos de una colección.

Cree documentos de muestra

```
db.users.insertMany([
  { name: "John", age: 30 },
  { name: "Jane", age: 25 },
  { name: "Bob", age: 35 },
  { name: "Alice", age: 28 }
]);
```

Ejemplo de consulta

```
db.users.aggregate([
  { $count: "total" }
]);
```

Salida

```
{ "total" : 4 }
```

El ejemplo agrega la `users` colección y usa el `$count` escenario para contar el número total de documentos.

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$count` comando, elija la pestaña correspondiente al idioma que desee utilizar:

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 (acumulador)

Nuevo a partir de la versión 8.0.1.

Utilice el `$count` acumulador dentro de una `$group` etapa para obtener el número de documentos de cada grupo. Acepta un objeto vacío `{}` como argumento.

Esto es diferente de la fase de `$count` tramitación, que es una fase independiente en la que se cuentan todos los documentos que pasan por la tramitación. En cambio, el `$count` acumulador cuenta los documentos de cada grupo producido por. `$group`

Sintaxis

```
{ $count: {} }
```

Parámetros

- El `$count` acumulador no acepta argumentos. Acepta un objeto vacío `{}` y devuelve el recuento de documentos de cada grupo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$count` acumulador para contar el número de productos de cada categoría.

Crea documentos de muestra

```
db.products.insertMany([
  { name: "Widget", category: "A" },
  { name: "Gadget", category: "A" },
  { name: "Doohickey", category: "B" },
  { name: "Thingamajig", category: "B" },
  { name: "Whatsit", category: "B" }
])
```

Ejemplo de consulta

```
db.products.aggregate([
  { $group: { _id: "$category", count: { $count: {} } } }
])
```

Salida

```
[
  { "_id": "A", "count": 2 },
  { "_id": "B", "count": 3 }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$count` acumulador también se puede utilizar como operador de ventanas en el escenario. `$setWindowFields` En este contexto, devuelve el número de documentos de cada ventana.

Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

Cree documentos de muestra

```
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 }
])
```

Ejemplo de consulta

En el ejemplo siguiente se dividen los documentos`category`, se ordenan las `day` particiones y se devuelve un recuento continuo de los documentos desde el principio de la partición hasta el documento actual.

```
db.events.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$category",
      sortBy: { day: 1 },
      output: {
        runningCount: {
          $count: {},
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
])
```

Salida

```
[
  { "_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 }
```

```
]
```

Cada documento se complementa con `runningCount` el número de documentos de su partición, hasta incluir el documento actual.

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$count` acumulador, elija la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran tanto el uso del acumulador (en`$group`) como el uso del operador de ventanas (en): `$setWindowFields`

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);

  try {
    await client.connect();

    const db = client.db('test');

    // Accumulator usage: count per group
    const products = db.collection('products');
    await products.insertMany([
      { name: "Widget", category: "A" },
      { name: "Gadget", category: "A" },
      { name: "Doohickey", category: "B" },
      { name: "Thingamajig", category: "B" },
      { name: "Whatsit", category: "B" }
    ]);
    const accumulatorResult = await products.aggregate([
      { $group: { _id: "$category", count: { $count: {} } } }
    ]).toArray();
    console.log('Accumulator result:', accumulatorResult);

    // Window operator usage: running count within each partition
    const events = db.collection('events');
    await events.insertMany([
```

```
      { _id: 1, category: "A", day: 1 },
      { _id: 2, category: "A", day: 2 },
      { _id: 3, category: "A", day: 3 },
      { _id: 4, category: "B", day: 1 },
      { _id: 5, category: "B", day: 2 }
    ]);
    const windowResult = await events.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$category",
          sortBy: { day: 1 },
          output: {
            runningCount: {
              $count: {},
              window: { documents: ["unbounded", "current"] }
            }
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);

  } catch (error) {
    console.error('Error:', error);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient
from pprint import pprint

def example():
    client = None
    try:
        client = MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
```

```
        db = client['test']

        # Accumulator usage: count per group
        products = db['products']
        products.insert_many([
            { 'name': 'Widget', 'category': 'A' },
            { 'name': 'Gadget', 'category': 'A' },
            { 'name': 'Doohickey', 'category': 'B' },
            { 'name': 'Thingamajig', 'category': 'B' },
            { 'name': 'Whatsit', 'category': 'B' }
        ])
        accumulator_result = list(products.aggregate([
            { '$group': { '_id': '$category', 'count': { '$count': {} } } }
        ]))
        pprint(accumulator_result)

        # Window operator usage: running count within each partition
        events = db['events']
        events.insert_many([
            { '_id': 1, 'category': 'A', 'day': 1 },
            { '_id': 2, 'category': 'A', 'day': 2 },
            { '_id': 3, 'category': 'A', 'day': 3 },
            { '_id': 4, 'category': 'B', 'day': 1 },
            { '_id': 5, 'category': 'B', 'day': 2 }
        ])
        window_result = list(events.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$category',
                    'sortBy': { 'day': 1 },
                    'output': {
                        'runningCount': {
                            '$count': {},
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
                    }
                }
            }
        ]))
        pprint(window_result)

    except Exception as e:
        print(f"An error occurred: {e}")
```

```
    finally:
        if client:
            client.close()

example()
```

$covariancePop

Nuevo a partir de la versión 8.0.2.

El `$covariancePop` operador de Amazon DocumentDB devuelve la covarianza poblacional de dos expresiones numéricas. Es un operador de ventana que solo se usa en la `$setWindowFields` etapa; no es válido en las etapas o. `$group` `$bucket` El operador se especifica en el `output` campo de la etapa y, si lo desea, puede definir los límites de la ventana con un `window` documento.

Parámetros

- `$covariancePop`toma una matriz de dos elementos`[ <expression1>, <expression2> ]`, donde cada elemento es una expresión numérica (una ruta de campo, por ejemplo`"$x"`, una expresión o una constante). Devuelve la covarianza poblacional de las dos expresiones en los documentos de la ventana.

Ejemplo (MongoDB Shell)

En el ejemplo siguiente se dividen los documentos`series`, se ordenan las particiones y se calcula la covarianza poblacional de toda la partición `x` y `y` de ella. `t`

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.measurements.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$series",
      sortBy: { t: 1 },
      output: {
        covariance: {
          $covariancePop: [ "$x", "$y" ],
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Todos los documentos de una partición reciben lo mismo`covariance`, es decir, la covarianza de `x` población de `y` todos los documentos de esa partición: `2` para las series `A` (`x` e `y` aumentan juntas) y `-2` para las series `B` (`y` disminuye a medida que `x` aumenta).

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$covariancePop` del operador en el `$setWindowFields` escenario, elija la pestaña correspondiente al idioma que desee utilizar:

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()
```

$covarianceSamp

Nuevo a partir de la versión 8.0.2.

El `$covarianceSamp` operador de Amazon DocumentDB devuelve la covarianza de muestra de dos expresiones numéricas. Es un operador de ventana que solo se usa en la `$setWindowFields` etapa; no es válido en las etapas o. `$group` `$bucket` El operador se especifica en el `output` campo de la etapa y, si lo desea, puede definir los límites de la ventana con un `window` documento.

Parámetros

- `$covarianceSamp`toma una matriz de dos elementos`[ <expression1>, <expression2> ]`, donde cada elemento es una expresión numérica (una ruta de campo, por ejemplo`"$x"`, una expresión o una constante). Devuelve el ejemplo de covarianza de las dos expresiones en los documentos de la ventana.

Ejemplo (MongoDB Shell)

En el ejemplo siguiente se dividen los documentos`series`, se ordenan las particiones y se calcula la covarianza muestral de `x` toda la partición. `t` y

Cree documentos de ejemplo

```
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 }
]);
```

Ejemplo de consulta

```
db.measurements.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$series",
      sortBy: { t: 1 },
      output: {
        covariance: {
```

```
          $covarianceSamp: [ "$x", "$y" ],
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Todos los documentos de una partición reciben la misma `covariance` covarianza muestral de `x` y todos los documentos de esa partición: `3` para las series `A` (`x` e `y` aumentan juntas) y `-3` para las series `B` (`y` disminuye a medida que `x` aumenta). La covarianza muestral se divide por, `n - 1` en lugar den, por lo que su magnitud es mayor que la covarianza poblacional de los mismos datos.

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$covarianceSamp` operador en el `$setWindowFields` escenario, elija la pestaña correspondiente al idioma que desee utilizar:

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 etapa de `$currentOp` agregación devuelve información sobre las operaciones que se están ejecutando actualmente en la base de datos. Esta etapa es útil para supervisar las consultas y operaciones activas en una canalización de agregación.

Parámetros

- `allUsers`(opcional): si se establece en`true`, devuelve las operaciones de todos los usuarios. El valor predeterminado es `false`.
- `idleConnections`(opcional): si se establece en`true`, incluye las conexiones inactivas. El valor predeterminado es `false`.
- `idleCursors`(opcional): cuando se establece en`true`, incluye información sobre los cursores inactivos. El valor predeterminado es `false`.
- `idleSessions`(opcional): si se establece en`true`, incluye información sobre las sesiones inactivas. El valor predeterminado es `true`.
- `localOps`(opcional): si se establece en`true`, incluye las operaciones locales. El valor predeterminado es `false`.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso de la etapa de `$currentOp` agregación para recuperar información sobre las operaciones de lectura activas.

Ejemplo de consulta

```
db.aggregate([
  { $currentOp: { allUsers: true, idleConnections: false } },
  { $match: { op: "query" } }
])
```

Salida

```
[
  {
    "opid": "12345",
    "active": true,
    "op": "query",
    "ns": "test.users",
    "secs_running": 2
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$currentOp` comando, elija la pestaña del idioma que desee usar:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('admin');

  const result = await db.aggregate([
    { $currentOp: { allUsers: true, idleConnections: false } },
    { $match: { op: "query" } }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['admin']

    result = list(db.aggregate([
        { '$currentOp': { 'allUsers': True, 'idleConnections': False } },
        { '$match': { 'op': 'query' } }
    ]))

    print(result)
    client.close()

example()
```

$dateAdd

Novedades de la versión 5.0

El operador de `$dateAdd` agregación de Amazon DocumentDB le permite añadir una duración a un valor de fecha y hora.

Parámetros

- `date`: un valor de fecha y hora al que añadir una duración.
- `duration`: la duración que se va a añadir al `date` valor. Se puede especificar como un objeto con claves para `years`,`months`, `weeks`,`days`,`hours`, `minutes`, `seconds`.
- `timezone`: (opcional) La zona horaria que se utilizará al realizar la suma de fechas. Si no se especifica, se utiliza la zona horaria predeterminada del clúster de Amazon DocumentDB.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$dateAdd` operador para añadir 2 días y 12 horas a una fecha.

Cree documentos de muestra

```
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") }
]);
```

Ejemplo de consulta

```
db.events.aggregate([
  {
    $project: {
      _id: 1,
      eventDate: 1,
      eventDatePlustwodaysandtwelvehours: {
        $dateAdd: {
          startDate: {
            $dateAdd: {
              startDate: "$eventDate",
```

```
              unit: "day",
              amount: 2
            }
          },
          unit: "hour",
          amount: 12
        }
      }
    }
  }
])
```

Salida

```
[
  {
    "_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)"
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$dateAdd` comando, elija la pestaña del idioma que desee usar:

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

Novedades de la versión 5.0

No es compatible con el clúster de Elastic.

El operador de `$dateDiff` agregación calcula la diferencia entre dos fechas en unidades específicas. Devuelve el número de límites de unidades cruzados entre las fechas de inicio y finalización.

Parámetros

- `startDate`: la expresión de fecha de inicio.
- `endDate`: La expresión de fecha final.
- `unit`: La unidad de tiempo de la diferencia. Las unidades compatibles son `year` `quarter` `month`, `week`, `day`, `hour` `minute`, `second`, y `millisecond`.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$dateDiff` operador para calcular el número de días entre la realización del pedido y la entrega.

Cree documentos de muestra

```
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")
  }
]);
```

Ejemplo de consulta

```
db.shipments.aggregate([
  {
    $project: {
      orderId: 1,
      orderDate: 1,
      deliveryDate: 1,
      daysToDeliver: {
        $dateDiff: {
```

```
          startDate: "$orderDate",
          endDate: "$deliveryDate",
          unit: "day"
        }
      }
    }
  }
]);
```

Salida

```
[
  {
    _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
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$dateDiff` comando, elija la pestaña del idioma que desee usar:

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()
```

$ fecha FromParts

Nuevo desde las versiones 5.0.1 y 8.0

El operador de `$dateFromParts` agregación de Amazon DocumentDB crea y devuelve un valor de fecha y hora a partir de sus partes constitutivas, como el año, el mes, el día, la hora, el minuto, el segundo y el milisegundo.

Parámetros

- `year`: El año natural. Necesario cuando no se utilizan piezas de la norma ISO 8601.
- `month`,`day`,`hour`,`minute`,`second`,`millisecond`: (opcional) Los componentes de fecha y hora restantes. Los componentes omitidos utilizan por defecto su valor válido más bajo.
- `isoWeekYear`,`isoWeek`,`isoDayOfWeek`: (opcional) los componentes de fecha según la norma ISO 8601. Utilícelos en lugar de `year`, `month`, y `day` para construir una fecha a partir de una fecha semanal según la norma ISO 8601.
- `timezone`: (opcional) La zona horaria que se utilizará al crear la fecha. Si no se especifica, se utiliza UTC.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$dateFromParts` operador para construir una fecha a partir de componentes individuales.

Cree documentos de muestra

```
db.parts.insertMany([
  { _id: 1, y: 2023, m: 4, d: 1 },
  { _id: 2, y: 2023, m: 12, d: 25 }
]);
```

Ejemplo de consulta

```
db.parts.aggregate([
  {
    $project: {
      _id: 1,
      constructedDate: {
        $dateFromParts: { year: "$y", month: "$m", day: "$d" }
      }
    }
  }
])
```

Salida

```
[
  { "_id": 1, "constructedDate": ISODate("2023-04-01T00:00:00Z") },
  { "_id": 2, "constructedDate": ISODate("2023-12-25T00:00:00Z") }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$dateFromParts` comando, elija la pestaña del idioma que desee usar:

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()
```

$ fecha FromString

El operador de `$dateFromString` agregación de Amazon DocumentDB le permite convertir una cadena de fecha y hora en un objeto de fecha. Esto resulta útil cuando los datos se almacenan como cadenas, pero necesita realizar operaciones basadas en fechas con los datos.

Parámetros

- `dateString`: cadena que representa una fecha y una hora.
- `format`: (opcional) Una cadena que especifica el formato de`dateString`. Si no se proporciona, Amazon DocumentDB intentará analizar la cadena en ese formato. ISO-8601
- `timezone`: (opcional) Una cadena que especifica la zona horaria. Si no se proporciona, Amazon DocumentDB utilizará la zona horaria del servidor.
- `onError`: (opcional) Especifica la acción que se debe realizar en caso de que se produzca un error en la conversión. Los valores posibles son `'error'` (el valor predeterminado, que genera un error), `'null'` (devuelve`null`) o `'replace'` (reemplaza el valor por la cadena de reemplazo especificada en la `onErrorMessage` opción).
- `onErrorMessage`: (opcional) Si `onError` se establece en`'replace'`, esta opción especifica la cadena de reemplazo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo `$dateFromString` convertir una cadena de fecha en un objeto de fecha en Amazon DocumentDB.

Cree documentos de muestra

```
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"}
]);
```

Ejemplo de consulta

```
db.missionLog.aggregate([
  {
    $project: {
      event: '$event',
      logDate: {
        $dateFromString: {
          dateString: '$logDate'
        }
      }
    }
  }
]);
```

Salida

```
[
  {
    "_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")
  }
```

```
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$dateFromString` comando, elija la pestaña del idioma que desee usar:

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()
```

$dateSubtract

Novedades de la versión 5.0

El operador de `$dateSubtract` agregación de Amazon DocumentDB le permite restar una duración específica de un valor de fecha.

Parámetros

- `date`: expresión de fecha que se convierte en una fecha o una marca de tiempo.
- `subtrahend`: una expresión de duración que especifica la cantidad de tiempo que se va a restar de la expresión. `date`
- `unit`: cadena que especifica la unidad de tiempo de la `subtrahend` expresión. Las unidades admitidas son «año», «trimestre», «mes», «semana», «día», «hora», «minuto», «segundo» y «milisegundo».

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$dateSubtract` operador para calcular la fecha de hace un año a partir de la fecha actual.

Cree documentos de muestra

```
db.events.insertOne({
  eventName: "Player joined",
  eventTime: ISODate("2023-04-01T12:00:00Z")
});
```

Ejemplo de consulta

```
db.events.aggregate([
  {
    $project: {
      eventName: 1,
      oneYearAgo: {
        $dateSubtract: {
          startDate: "$eventTime",
          amount: 1,
          unit: "year"
        }
      }
    }
  }
])
```

Salida

```
{
  "_id" : ObjectId("64567890abcdef012345678"),
  "eventName" : "Player joined",
  "oneYearAgo" : ISODate("2022-04-01T12:00:00Z")
}
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$dateSubtract` comando, elija la pestaña del idioma que desee usar:

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()
```

$ fecha ToParts

Nuevo desde las versiones 5.0.1 y 8.0

El operador de `$dateToParts` agregación de Amazon DocumentDB devuelve un documento que contiene las partes constitutivas de un valor de fecha y hora, como el año, el mes, el día, la hora, el minuto, el segundo y el milisegundo.

Parámetros

- `date`: El valor de fecha y hora que se va a dividir en partes.
- `timezone`: (opcional) La zona horaria que se utilizará al extraer las partes de fecha. Si no se especifica, las partes se devuelven en UTC.
- `iso8601`: (opcional) Si se establece en`true`, el documento devuelto utiliza partes de fecha ISO 8601 (`isoWeekYear`,`isoWeek`,`isoDayOfWeek`). El valor predeterminado es `false`.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$dateToParts` operador para devolver las partes individuales de una fecha.

Cree documentos de muestra

```
db.events.insertMany([
  { _id: 1, eventDate: ISODate("2023-04-01T10:30:15Z") },
  { _id: 2, eventDate: ISODate("2023-12-25T18:45:00Z") }
]);
```

Ejemplo de consulta

```
db.events.aggregate([
  {
    $project: {
      _id: 1,
      parts: { $dateToParts: { date: "$eventDate" } }
    }
  }
])
```

Salida

```
[
  {
    "_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 }
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$dateToParts` comando, elija la pestaña del idioma que desee usar:

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()
```

$ fecha ToString

El operador de `$dateToString` agregación de Amazon DocumentDB se utiliza para convertir un valor de fecha o marca horaria en una representación de cadena. Esto resulta útil cuando necesita formatear la fecha y la hora de una manera específica para mostrarlas o procesarlas posteriormente.

Parámetros

- `date`: el valor de fecha o marca horaria que se va a convertir en una cadena.
- `format`: cadena que especifica el formato en el que debe representarse la fecha. La cadena de formato puede incluir varios especificadores de formato, por ejemplo, `%Y` para el año de cuatro dígitos, `%m` para el mes de dos dígitos, `%d` para el día del mes de dos dígitos, etc.
- `timezone`: (opcional) La zona horaria que se va a utilizar para la conversión. Si no se especifica, se utiliza la zona horaria del servidor que aloja el clúster de Amazon DocumentDB.
- `onNull`: (opcional) El valor que se devolverá si el `date` parámetro es`null`.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$dateToString` operador para formatear el `logDate` campo de la `missionLog` colección.

Cree documentos de muestra

```
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") }
```

```
]);
```

Ejemplo de consulta

```
db.missionLog.aggregate([
  {
    $project: {
      event: "$event",
      logDateFormatted: {
        $dateToString: {
          format: "%Y-%m-%d %H:%M:%S",
          date: "$logDate"
        }
      }
    }
  }
])
```

Salida

```
[
  {
    "_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"
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$dateToString` comando, elija la pestaña del idioma que desee usar:

Node.js

A continuación, se muestra un ejemplo del uso `$dateToString` del operador en una Node.js aplicación:

```
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

Este es un ejemplo del uso del `$dateToString` operador en una aplicación de Python:

```
from pymongo 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

Novedades de la versión 8.0

No es compatible con el clúster de Elastic.

El operador de `$dateTrunc` agregación de Amazon DocumentDB trunca una fecha a una unidad específica.

Parámetros

- `date`: expresión de fecha que se convierte en una fecha o una marca de tiempo.
- `unit`: cadena que especifica la unidad de tiempo de la expresión de sustracción. Las unidades admitidas son `yearquarter`,`month`,`week`,`day`, `hour` `minutesecond`, y. `millisecond`

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$dateTrunc` operador para truncar una fecha y convertirla en una hora.

Cree documentos de muestra

```
db.events.insertMany([
  {
    eventName: "Event 1",
    eventTime: ISODate("2025-04-01T12:15:00Z")
  },
  {
    eventName: "Event 2",
    eventTime: ISODate("2025-08-15T14:33:22Z")
  },
]);
```

Ejemplo de consulta

```
db.events.aggregate([
  {
    $project: {
      eventName: 1,
      eventTime: 1,
      truncatedToHour: {
        $dateTrunc: {
          date: "$eventTime",
          unit: "hour"
        }
      }
    }
  }
]);
```

Salida

```
[
  {
    _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')
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$dateTrunc` comando, elija la pestaña del idioma que desee usar:

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()
```

$día OfMonth

El operador de `$dayOfMonth` agregación de Amazon DocumentDB recupera el día del mes (del 1 al 31) de una fecha determinada. Este operador resulta útil para agrupar, filtrar o extraer el día del mes de los campos de fecha de los documentos.

Parámetros

- `date expression`: la expresión de fecha puede ser un campo de fecha del documento, un objeto de fecha o una cadena de fecha.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo utilizar el `$dayOfMonth` operador para extraer el día del mes de un campo de fecha del documento.

Cree documentos de muestra

```
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") }
]);
```

Ejemplo de consulta

```
db.events.aggregate([
  { $project: {
    eventDay: { $dayOfMonth: "$eventDate" }
  }}
])
```

Salida

```
[
  { "_id" : 1, "eventDay" : 15 },
  { "_id" : 2, "eventDay" : 28 },
  { "_id" : 3, "eventDay" : 1 },
  { "_id" : 4, "eventDay" : 30 }
```

```
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$dayOfMonth` comando, elija la pestaña del idioma que desee usar:

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()
```

$día OfWeek

El `$dayOfWeek` operador de Amazon DocumentDB extrae el día de la semana de un campo de fecha determinado. Devuelve el día de la semana como un número entre 1 (domingo) y 7 (sábado), que es el mismo comportamiento que en MongoDB.

Parámetros

- `date field`: el campo de fecha del que se extrae el día de la semana.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo utilizar el `$dayOfWeek` operador para extraer el día de la semana del `date` campo de la colección. `weather`

Cree documentos de muestra

```
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")
```

```
  }
]);
```

Ejemplo de consulta

```
db.weather.aggregate([
  {
    $project: {
      dayOfWeek: { $dayOfWeek: "$date" }
    }
  }
]).pretty();
```

Salida

```
{ "_id" : ObjectId("64272c6663f4f8ce422c2d91"), "dayOfWeek" : 7 }
{ "_id" : ObjectId("64272c6663f4f8ce422c2d92"), "dayOfWeek" : 1 }
{ "_id" : ObjectId("64272c6663f4f8ce422c2d93"), "dayOfWeek" : 2 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$dayOfWeek` comando, elija la pestaña del idioma que desee usar:

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()
```

$día OfYear

El `$dayOfYear` operador de Amazon DocumentDB devuelve el día del año de una fecha como un número entre 1 y 366 (365 en los años no bisiestos).

Parámetros

- `expression`: el campo o expresión de fecha del que se extrae el día del año.

Ejemplo (MongoDB Shell)

En este ejemplo, se muestra cómo utilizar el `$dayOfYear` operador para extraer el día del año de un campo de fecha de una colección de Amazon DocumentDB.

Cree documentos de muestra

```
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")
  }
])
```

Ejemplo de consulta

```
db.weather.aggregate([
  {
    $project: {
      dayOfYear: { $dayOfYear: "$date" }
    }
  }
]).pretty()
```

Salida

```
{ "_id" : ObjectId("642b86fc7d8e07af279bbe63"), "dayOfYear" : 74 }
{ "_id" : ObjectId("642b86fc7d8e07af279bbe64"), "dayOfYear" : 365 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$dayOfYear` comando, elija la pestaña del idioma que desee usar:

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()
```

$ grados ToRadians

Nuevo a partir de la versión 8.0.1.

El `$degreesToRadians` operador de Amazon DocumentDB convierte un valor de grados a radianes.

Parámetros

- `expression`: expresión que se resuelve en un valor numérico en grados.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$degreesToRadians` operador para convertir valores de grados en radianes.

Cree documentos de muestra

```
db.angles.insertMany([
  { "_id": 1, "degrees": 0 },
  { "_id": 2, "degrees": 90 },
  { "_id": 3, "degrees": 180 },
  { "_id": 4, "degrees": 360 }
]);
```

Ejemplo de consulta

```
db.angles.aggregate([
  { $project: {
    "radians": { $degreesToRadians: "$degrees" }
  }}
]);
```

Salida

```
[
  { "_id": 1, "radians": 0 },
  { "_id": 2, "radians": 1.5707963267948966 },
  { "_id": 3, "radians": 3.141592653589793 },
  { "_id": 4, "radians": 6.283185307179586 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$degreesToRadians` operador, elige la pestaña del idioma que quieres usar:

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

Nuevo a partir de la versión 8.0.2.

El operador de `$denseRank` ventanas de Amazon DocumentDB devuelve la posición de cada documento dentro de su partición, según el orden de las `$setWindowFields` etapas`sortBy`. Los documentos con el mismo `sortBy` valor reciben la misma clasificación y el siguiente valor distinto utiliza la posición inmediatamente posterior, sin dejar espacios en blanco (por ejemplo,`1, 2, 2, 3`). Se especifica en el `output` campo del `$setWindowFields` escenario.

Es un operador de ventana que se usa solo en el `$setWindowFields` escenario.

Parámetros

- `$denseRank`toma `{}` como valor un documento vacío.
- No acepta `window` marcos.
- Se necesita exactamente una `sortBy` tecla en el `$setWindowFields` escenario.

Ejemplo (MongoDB Shell)

En el ejemplo siguiente se dividen los documentos`game`, se ordenan las particiones `score` en orden descendente y se asigna a cada jugador una clasificación densa.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.players.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$game",
      sortBy: { score: -1 },
      output: {
        denseRank: { $denseRank: {} }
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Dentro de cada `game` partición, los jugadores se clasifican de forma descendente. `score` Bob y Carol empatan `85` y comparten rango`2`, y Dave sigue en rango `3` sin diferencia.

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$denseRank` operador en el `$setWindowFields` escenario, seleccione la pestaña correspondiente al idioma que desee utilizar:

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

El `$divide` operador de la canalización de agregación de Amazon DocumentDB se utiliza para dividir un número por otro. Es un operador útil para realizar operaciones matemáticas en campos numéricos de sus documentos.

Parámetros

- `numerator`: El dividendo o el número a dividir.
- `denominator`: El divisor o el número por el que se va a dividir.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo utilizar el `$divide` operador para calcular la tarifa horaria de los empleados en función de su salario anual y del número de horas de trabajo al año.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.employees.aggregate([
  {
    $project: {
      name: 1,
      hourlyRate: { $divide: ["$salary", "$hoursPerYear"] }
    }
  }
])
```

Salida

```
[
  { "_id" : 1, "name" : "John Doe", "hourlyRate" : 38.46153846153846 },
  { "_id" : 2, "name" : "Jane Smith", "hourlyRate" : 43.26923076923077 },
  { "_id" : 3, "name" : "Bob Johnson", "hourlyRate" : 36.05769230769231 }
```

```
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$divide` comando, elija la pestaña del idioma que desee usar:

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()
```

$documentNumber

Nuevo a partir de la versión 8.0.2.

El operador de `$documentNumber` ventanas de Amazon DocumentDB devuelve un número secuencial distinto para cada documento de su partición, según el orden de las `$setWindowFields` etapas. `sortBy` A diferencia de `$rank` y`$denseRank`, no trata los vínculos de forma especial: cada documento recibe un número único (por ejemplo,`1, 2, 3, 4`) y los documentos con `sortBy` valores iguales se numeran en el orden en que aparecen. Se especifica en el `output` campo de la `$setWindowFields` fase.

Es un operador de ventana que se usa solo en el `$setWindowFields` escenario.

Parámetros

- `$documentNumber`toma `{}` como valor un documento vacío.
- No acepta `window` marcos.
- Se necesita exactamente una `sortBy` tecla en el `$setWindowFields` escenario.

Ejemplo (MongoDB Shell)

En el ejemplo siguiente se dividen los documentos`game`, se ordenan las particiones `score` en orden descendente y se asigna a cada jugador un número de documento secuencial.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.players.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$game",
      sortBy: { score: -1 },
      output: {
        documentNumber: { $documentNumber: {} }
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Dentro de cada `game` partición, cada jugador recibe un número secuencial distinto en orden descendente. `score` Bob y Carol empatan 85 pero siguen numerados 2 y 3 en el orden en que aparecen.

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$documentNumber` operador en el `$setWindowFields` escenario, seleccione la pestaña correspondiente al idioma que desee utilizar:

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

El operador de `$eq` agregación compara dos valores y devuelve `true` si son iguales; de lo contrario, devuelve`false`.

Parámetros

- `expression1`: El primer valor que se va a comparar.
- `expression2`: El segundo valor que se va a comparar.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra el uso del `$eq` operador para comprobar si las cantidades de producto coinciden con los valores objetivo.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.inventory.aggregate([
  {
    $project: {
      item: 1,
      qty: 1,
      target: 1,
      meetsTarget: { $eq: ["$qty", "$target"] }
    }
  }
]);
```

Salida

```
[
  { _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 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el operador de $eq agregación, elija la pestaña correspondiente al idioma que desee usar:

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

Novedades de la versión 4.0

El `$exp` operador de Amazon DocumentDB le permite elevar la constante e a un número determinado.

Parámetros

- `expression`: La expresión que se va a evaluar. Puede ser cualquier expresión de agregación válida, incluidas las referencias de campo, las operaciones aritméticas y otras etapas de agregación.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$exp` operador para buscar todos los documentos en los que el `quantity` campo es mayor que el `price` campo.

Cree documentos de muestra

```
db.items.insertMany([
  { item: "canvas", quantity: 4 },
  { item: "journal", quantity: 2 }
]);
```

Ejemplo de consulta

```
db.items.aggregate([
  { $project: {
    "quantityRaised": {$exp: "$quantity"}}
  }
]);
```

Salida

```
[
  {
    _id: ObjectId('6920b785311cf98b79d2950d'),
    quantityRaised: 54.598150033144236
  },
  {
    _id: ObjectId('6920b785311cf98b79d2950e'),
    quantityRaised: 7.38905609893065
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$exp` comando, elija la pestaña del idioma que desee usar:

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

Nuevo a partir de la versión 8.0.2.

La fase de `$facet` agregación de Amazon DocumentDB procesa varias canalizaciones de agregación en una sola etapa en el mismo conjunto de documentos de entrada. Cada campo de salida define su propia canalización y cada canalización recibe los mismos documentos de entrada. Esto le permite calcular varias agregaciones independientes, como recuentos, agrupaciones y

estadísticas resumidas, en una sola consulta, en lugar de ejecutar la recopilación a través de varias consultas independientes.

Cada subcanalización se ejecuta de forma independiente y produce una matriz de documentos que se almacena en su campo de salida correspondiente. Como las subcanalizaciones no comparten el estado, la salida de una canalización no afecta a la otra.

Sintaxis

```
{
  $facet: {
    <outputField1>: [ <stage1>, <stage2>, ... ],
    <outputField2>: [ <stage1>, <stage2>, ... ],
    ...
  }
}
```

Parámetros

- `outputField`(obligatorio): el nombre de un campo de salida que contiene la matriz de resultados de su canalización de agregación asociada. Puede especificar uno o más campos de salida. Cada campo de salida se asigna a una matriz de etapas del proceso de agregación que se aplica a los documentos de entrada.

Salida

La `$facet` etapa devuelve un único documento. Cada campo de salida que especifique contiene una matriz de documentos producidos por su subcanalización correspondiente.

Limitaciones

- Una `$facet` subcanalización no puede incluir las siguientes etapas:`$changeStream`,`$collStats`, `$facet` (no se permite el anidamiento),,`$geoNear`,`$indexStats`,`$listSearchIndexes`,`$merge`,, `$out` `$planCacheStats$search`, `$searchMeta` y. `$vectorSearch`
- El documento que devuelva la `$facet` etapa debe mantenerse dentro del límite de tamaño del documento BSON de 16 MB.
- Una `$sort` etapa que precede a una `$facet` etapa no puede determinar el orden de un acumulador sensible a los pedidos (`$first`,,`$last`, `$push` `$firstN``$lastN`, o`$mergeObjects`)

en una $bucketAuto etapa $group$bucket, o en la que ese pedido tendría que aplicarse más allá del límite. $facet Esto se aplica tanto si la fase de agrupamiento se encuentra dentro de una $facet subcanalización como después de la fase. $facet En este caso, Amazon DocumentDB devuelve un error en lugar de ejecutar la canalización. Para pedir un acumulador de este tipo, coloque la $sort etapa dentro de la $facet subcanalización, inmediatamente antes de la etapa de agrupamiento. Order-insensitive los acumuladores, como,, y,$sum, minmax, no se ven afectados. $count

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el $facet escenario para calcular varias agregaciones de los mismos datos de ventas en una sola consulta.

Cree documentos de muestra

```
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") }
]);
```

Ejemplo de consulta

El siguiente comando se utiliza $facet para calcular, en una sola etapa, la cantidad total vendida por artículo, las estadísticas generales de precios y el recuento total de documentos:

```
db.sales.aggregate([
  {
    $facet: {
      "totalByItem": [
        { $group: { _id: "$item", totalQuantity: { $sum: "$quantity" } } }
      ],
      "priceStats": [
        { $group: { _id: null, avgPrice: { $avg: "$price" }, maxPrice: { $max:
 "$price" } } }
      ],
      "documentCount": [
        { $count: "total" }
      ]
```

```
    }
  }
])
```

Salida

El comando devuelve el siguiente resultado:

```
[
  {
    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 }
    ]
  }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$facet` comando, elige la pestaña del idioma que quieres usar:

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

El `$filter` operador de Amazon DocumentDB se usa para aplicar una expresión de filtro a cada elemento de una matriz y devolver una matriz que contiene solo los elementos que cumplen con la condición especificada. Este operador es útil cuando necesita realizar operaciones de filtrado complejas en los campos de la matriz de los documentos.

Parámetros

- `input`: el campo de matriz que se va a filtrar.
- `as`: el nombre de la variable que se utilizará para cada elemento de la `input` matriz dentro de la `cond` expresión.
- `cond`: la expresión booleana que determina si un elemento determinado debe incluirse en la matriz de salida.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$filter` operador para proyectar el cliente de cada pedido y crear un nuevo campo de matriz paidItems que contenga solo los artículos de la matriz de artículos cuyo precio es superior a 15. Básicamente, filtra los artículos de cada pedido para incluir solo los productos que cuestan más de 15.

Crea documentos de muestra

```
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 }
  ]}
]);
```

Ejemplo de consulta

```
db.orders.aggregate([
  {
    $project: {
      customer: 1,
      paidItems: {
        $filter: {
          input: "$items",
          as: "item",
          cond: { $gt: ["$$item.price", 15] }
        }
      }
    }
  }
]).pretty();
```

Salida

```
[
  {
    _id: 1,
    customer: 'abc123',
    paidItems: [ { name: 'Product B', price: 20, qty: 1 } ]
  },
  { _id: 2, customer: 'def456', paidItems: [] },
  { _id: 3, customer: 'ghi789', paidItems: [] }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$filter` comando, elige la pestaña del idioma que quieres usar:

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

Nuevo a partir de la versión 5.0.

Elastic Cluster no lo admite.

El `$first` operador de Amazon DocumentDB devuelve el primer documento de un conjunto de documentos agrupados. Se utiliza habitualmente en las canalizaciones de agregación para recuperar el primer documento que coincide con una condición específica.

Parámetros

- `expression`: la expresión que se devuelve como primer valor de cada grupo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$first` operador para recuperar el primer valor de elemento encontrado para cada categoría durante la agregación.

Nota: `$first` devuelve el primer documento según el orden actual de los documentos en proceso. Para garantizar un orden específico (por ejemplo, por fecha, precio, etc.), se debe utilizar una `$sort` fase antes de la `$group` fase.

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.products.aggregate([
  { $group: { _id: "$category", firstItem: { $first: "$item" } } }
]);
```

Salida

```
[
  { "_id" : "food", "firstItem" : "abc" },
  { "_id" : "toy", "firstItem" : "xyz" }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$first` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve el valor del `expression` del primer documento de cada ventana. Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

> **Note**
>
> Cuando se usa como operador de ventana en`$setWindowFields`, `$first` está limitado a 100 MB de datos intermedios. Una operación que supera este límite devuelve un error.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

En el ejemplo siguiente se dividen los documentos`ticker`, se ordenan las `hour` particiones y se devuelve el primer precio (el más antiguo) registrado en cada partición.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        firstPrice: {
          $first: "$price",
          window: { documents: ["unbounded", "current"] }
        }
```

```
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Cada documento se complementa con `firstPrice` el precio del primer documento de su ventana (el primero `hour` de la partición).

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$first` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran tanto el uso del acumulador (en`$group`) como el uso del operador de ventanas (en`$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()
```

$primeroN

Nuevo a partir de la versión 8.0.1.

El `$firstN` operador de Amazon DocumentDB devuelve los primeros N elementos. Cuando se usa como acumulador en una `$group` etapa, devuelve una matriz con los primeros N valores de cada grupo. Cuando se usa como operador de expresión matricial, devuelve los primeros N elementos de una matriz.

Parámetros

- `input`: la expresión que se convierte en el campo o la matriz desde los que se devuelven los valores.
- n: un entero positivo que especifica cuántos valores se devolverán. Cuando se usa como acumulador en una `$group` etapa, también n puede ser una expresión, siempre que se resuelva como un entero positivo según el campo del grupo`_id`.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$firstN` acumulador para recuperar las dos primeras cantidades de cada artículo durante la agregación.

> **Note**
>
> `$firstN`selecciona los valores en el orden en que los documentos llegan al `$group` escenario. Para devolver los primeros N valores de un orden específico (por ejemplo, por fecha o puntuación), añada una `$sort` etapa antes`$group`.

Crea documentos de muestra

```
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") }
]);
```

Ejemplo de consulta

```
db.sales.aggregate([
  { $group: { _id: "$item", firstTwoQuantities: { $firstN: { input: "$quantity", n:
 2 } } } }
]);
```

Salida

```
[
  { "_id": "abc", "firstTwoQuantities": [10, 5] },
  { "_id": "xyz", "firstTwoQuantities": [15, 7] }
]
```

Ejemplo de uso de expresiones (MongoDB Shell)

El `$firstN` operador también se puede usar como expresión dentro de una `$project` etapa para devolver los primeros N elementos de un campo matricial.

Crea documentos de muestra

```
db.inventory.insertMany([
  { _id: 1, item: "abc", tags: ["red", "green", "blue", "yellow", "purple"] },
  { _id: 2, item: "xyz", tags: ["alpha", "beta", "gamma"] }
]);
```

Ejemplo de consulta

```
db.inventory.aggregate([
  { $project: {
      firstThreeTags: { $firstN: { input: "$tags", n: 3 } }
    }}
]);
```

Salida

```
[
  { "_id": 1, "firstThreeTags": ["red", "green", "blue"] },
  { "_id": 2, "firstThreeTags": ["alpha", "beta", "gamma"] }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$firstN` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve una matriz con hasta los primeros n valores de los documentos de cada ventana. Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

> **Note**
>
> Cuando se usa como operador de ventana en`$setWindowFields`, `$firstN` está limitado a 100 MB de datos intermedios. Una operación que supera este límite devuelve un error.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

En el ejemplo siguiente se dividen los documentos`ticker`, se ordenan las `hour` particiones y se devuelven los dos primeros precios desde el principio de la partición hasta el documento actual.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        firstTwoSoFar: {
          $firstN: { input: "$price", n: 2 },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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] }
]
```

Cada documento se complementa con `firstTwoSoFar` los dos primeros `price` valores de su ventana (en `hour` orden).

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$firstN` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran el uso del acumulador (in`$group`), el uso de expresiones (in`$project`) y el uso del operador de ventana (in`$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

Nuevo desde la versión 4.0.

El `$floor` operador de Amazon DocumentDB devuelve el entero más grande que sea menor o igual al número especificado. Este operador es útil para redondear valores numéricos.

Parámetros

- `expression`: la expresión numérica que se va a redondear hacia abajo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$floor` operador para redondear un valor decimal al entero más cercano.

Cree documentos de muestra

```
db.numbers.insertOne({ value: 3.14 });
```

Ejemplo de consulta

```
db.numbers.aggregate([
  { $project: { _id: 0, floored: { $floor: "$value" } } }
]);
```

Salida

```
{ "floored" : 3 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$floor` comando, elija la pestaña del idioma que desee usar:

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 etapa de `$geoNear` agregación devuelve los documentos en orden de proximidad a un punto específico. Calcula la distancia desde el punto e incluye la distancia en los documentos de salida.

Parámetros

- `near`: el punto desde el que se calculan las distancias, especificado como GeoJSON o coordenadas heredadas.
- `distanceField`: el nombre del campo para almacenar la distancia calculada.
- `spherical`: booleano que indica si se debe utilizar geometría esférica (obligatorio para los puntos GeoJSON).
- `maxDistance`: opcional. Distancia máxima desde el punto central.
- `minDistance`: opcional. Distancia mínima desde el punto central.
- `query`: opcional. Criterios de filtro adicionales que se deben aplicar.
- `limit`: opcional. Número máximo de documentos a devolver.
- `key`: opcional. Campo que se utilizará para la consulta geoespacial cuando existen varios índices geoespaciales.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra el uso del `$geoNear` escenario para buscar las tiendas más cercanas a una ubicación determinada.

Cree documentos de muestra

```
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] } }
]);
```

Ejemplo de consulta

```
db.stores.aggregate([
  {
    $geoNear: {
      near: { type: "Point", coordinates: [-122.4, 37.8] },
      distanceField: "distance",
      spherical: true
    }
```

```
  }
]);
```

Salida

```
[
  { _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 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar la etapa de `$geoNear` agregación, elija la pestaña correspondiente al idioma que desee usar:

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

Nuevo a partir de la versión 8.0.2.

La fase de `$graphLookup` agregación de Amazon DocumentDB realiza una búsqueda recursiva en una colección. Para cada documento de entrada, `$graphLookup` comienza con uno o más valores y hace coincidir repetidamente un campo de conexión con un campo de destino, siguiendo la cadena de documentos coincidentes hasta que no haya más coincidencias o se alcance una profundidad de recursión opcional. Esto resulta útil para analizar datos jerárquicos o similares a gráficos, como las estructuras jerárquicas organizativas, los árboles de categorías o las relaciones de red.

Parámetros

- `from`: la colección de destino para buscar de forma recursiva. La `from` colección debe estar en la misma base de datos.
- `startWith`: la expresión que especifica el valor o los valores con los que iniciar la búsqueda recursiva. Este valor se compara con el `connectToField` de los documentos de la `from` colección. Si `startWith` se evalúa como una matriz, la búsqueda se ejecuta desde todos los elementos de la matriz al mismo tiempo.
- `connectFromField`: el nombre del campo cuyo valor se usa para conectarse al siguiente documento de la búsqueda recursiva. Si el valor es una matriz, cada elemento se sigue de forma individual a lo largo del recorrido.
- `connectToField`: el nombre del campo de la `from` colección que se compara con el valor de. `connectFromField`
- `as`: el nombre del campo de matriz agregado a cada documento de salida. Este campo contiene los documentos encontrados durante la búsqueda recursiva. Para obtener información sobre el orden de estos documentos, consulte el elemento Ordenamiento de los resultados en la siguiente sección Comportamiento.
- `maxDepth`: un entero no negativo que especifica el número máximo de iteraciones recursivas que se deben realizar después de la búsqueda inicial. Este parámetro es opcional. Un valor de `0` realiza únicamente la coincidencia inicial y no la recursión. Si lo omite`maxDepth`, la búsqueda continúa de forma recursiva hasta que no haya más documentos coincidentes.
- `depthField`: el nombre del campo que se va a agregar a cada documento de la `as` matriz que contiene la profundidad de recursión con la que se encontró el documento. Este parámetro es opcional. La profundidad se representa como un `NumberLong` y comienza en cero, por lo que la búsqueda inicial corresponde a una profundidad de`0`.
- `restrictSearchWithMatch`: documento que especifica una condición de coincidencia adicional que se aplica a los documentos de la búsqueda. Este parámetro es opcional. La condición usa la misma sintaxis de consulta que en una `$match` etapa y no puede incluir expresiones de agregación.

> **Note**
>
> La `$graphLookup` etapa está sujeta a un límite de memoria de 100 MB y no es compatible`allowDiskUse:true`. Una operación que supera este límite devuelve un error. Si la agregación incluye otras etapas, la `allowDiskUse:true` opción permanece en vigor para esas etapas.

Comportamiento

- Cross-collection transversal: `from` defina la colección en la que desea buscar. Puede ser la misma colección en la que se ejecuta la canalización o una colección diferente en la misma base de datos.
- Orden de los resultados: los documentos de la `as` matriz no se devuelven en un orden garantizado.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se utiliza `$graphLookup` para rebasar la jerarquía jerárquica de los empleados y, para cada empleado, mostrar la cadena completa de gerentes por encima de ellos.

Los siguientes comandos, que se ejecutan en el shell mongosh, insertan ejemplos de documentos de empleados que establecen una jerarquía jerárquica:

```
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 siguiente agregación muestra la cadena de administración completa de cada empleado de una nueva `reportingHierarchy` matriz:

```
db.employees.aggregate([
  {
    $graphLookup: {
      from: "employees",
      startWith: "$reportsTo",
      connectFromField: "reportsTo",
      connectToField: "name",
      as: "reportingHierarchy",
      depthField: "level"
    }
  }
]);
```

Esta operación devuelve el siguiente resultado:

```
[
  {
    _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: []
```

```
  }
]
```

Cada documento del empleado obtiene una `reportingHierarchy` matriz que contiene los documentos del administrador coincidentes de forma recursiva. El `level` campo registra la profundidad de recursión a la que se encontró cada administrador. Asya no tiene ningún `reportsTo` valor, por lo que su `reportingHierarchy` matriz está vacía.

Ejemplo con maxDepth y restrict SearchWithMatch (MongoDB Shell)

El siguiente ejemplo recorre una red de conexiones entre personas y se utiliza `maxDepth` para limitar la recursión e `restrictSearchWithMatch` incluir solo los documentos que cumplen una condición adicional. La búsqueda busca las conexiones de una persona que está empezando, con una profundidad de hasta dos niveles, y solo conserva las personas cuya `hobbies` matriz contiene. `golf`

Los siguientes comandos, ejecutados en la consola mongosh, insertan documentos de ejemplo en los que cada persona `friends` señala hacia adelante a través de una cadena de conexiones unidireccional:

```
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 siguiente agregación parte de Tanya, recorre su red de conexiones hasta dos niveles de profundidad y solo incluye a las personas que la listan como pasatiempo: `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 } }
]);
```

Esta operación devuelve el siguiente resultado:

```
[
  {
    _id: 1,
    name: 'Tanya',
    golfConnections: [
      { name: 'Carole' },
      { name: 'Joseph' },
      { name: 'Angelo' }
    ]
  }
]
```

Solo `golf` se incluyen las conexiones cuya `hobbies` matriz contenga, y el recorrido se detiene después de dos niveles de recursión. Shirley está excluida porque no aparece en la lista`golf`, y Miguel está excluido porque está a tres niveles de Tanya, más allá del de. `maxDepth` 2 Como las conexiones forman una cadena unidireccional que nunca apunta a Tanya, el documento inicial no coincide durante la búsqueda recursiva.

Ejemplos de código

Para ver un ejemplo de código que crea la jerarquía de informes de los empleados a partir del primer ejemplo con un controlador MongoDB, seleccione la pestaña correspondiente al idioma que desee utilizar:

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 de `$group` agregación de Amazon DocumentDB le permite agrupar documentos según una expresión específica y realizar varias operaciones acumulativas en los datos agrupados. Esto puede resultar útil para tareas como el cálculo de totales, promedios u otras estadísticas basadas en los datos agrupados.

Parámetros

- `_id`: especifica la expresión con la que se deben agrupar los documentos de entrada. Puede ser un nombre de campo, una expresión calculada o una combinación de ambos.
- `accumulator expressions`: (opcional) Una o más expresiones acumuladoras que se deben aplicar a los datos agrupados. Estas expresiones utilizan los operadores acumuladores mencionados anteriormente.

Ejemplo (MongoDB Shell)

El siguiente ejemplo agrupa a los clientes por ciudad y calcula el importe total del pedido para cada ciudad.

Crea documentos de muestra

```
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 }] }
]);
```

Ejemplo de consulta

```
db.customers.aggregate([
  {
    $group: {
      _id: "$city",
      totalOrders: { $sum: { $sum: "$orders.amount" } }
    }
  }
]);
```

Salida

```
[
  { _id: 'Chicago', totalOrders: 150 },
  { _id: 'Los Angeles', totalOrders: 450 },
  { _id: 'New York', totalOrders: 500 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$group` comando, elige la pestaña del idioma que quieres usar:

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

El operador de `$gt` agregación compara dos valores y devuelve `true` si el primer valor es mayor que el segundo; de lo contrario, devuelve`false`.

Parámetros

- `expression1`: El primer valor que se va a comparar.
- `expression2`: El segundo valor que se va a comparar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$gt` operador para identificar los productos que superan un umbral de precio.

Cree documentos de muestra

```
db.products.insertMany([
  { _id: 1, name: "Laptop", price: 1200 },
  { _id: 2, name: "Mouse", price: 25 },
  { _id: 3, name: "Monitor", price: 400 }
]);
```

Ejemplo de consulta

```
db.products.aggregate([
```

```
  {
    $project: {
      name: 1,
      price: 1,
      expensive: { $gt: ["$price", 100] }
    }
  }
]);
```

Salida

```
[
  { _id: 1, name: 'Laptop', price: 1200, expensive: true },
  { _id: 2, name: 'Mouse', price: 25, expensive: false },
  { _id: 3, name: 'Monitor', price: 400, expensive: true }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el operador de $gt agregación, elija la pestaña correspondiente al idioma que desee usar:

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

El operador de `$gte` agregación compara dos valores y devuelve `true` si el primer valor es mayor o igual que el segundo; de lo contrario, devuelve`false`.

Parámetros

- `expression1`: el primer valor que se va a comparar.
- `expression2`: El segundo valor que se va a comparar.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra el uso del `$gte` operador para comprobar si los estudiantes han aprobado un examen.

Cree documentos de muestra

```
db.students.insertMany([
  { _id: 1, name: "Alice", score: 85 },
  { _id: 2, name: "Bob", score: 60 },
  { _id: 3, name: "Charlie", score: 72 }
]);
```

Ejemplo de consulta

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      score: 1,
      passed: { $gte: ["$score", 70] }
    }
  }
]);
```

Salida

```
[
  { _id: 1, name: 'Alice', score: 85, passed: true },
  { _id: 2, name: 'Bob', score: 60, passed: false },
  { _id: 3, name: 'Charlie', score: 72, passed: true }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el operador de `$gte` agregación, elija la pestaña correspondiente al idioma que desee usar:

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

El $hour operador extrae el componente de hora de un campo de fecha o marca de hora.

Parámetros

- dateExpression: la fecha a la que se aplica el operador. Esto debe resolverse en una fecha BSON válida (por ejemplo, un campo como $createdAt o una fecha literal).

Un parámetro también se puede especificar como un documento con el siguiente formato:

{fecha:<dateExpression>, zona horaria:} <timezoneExpression>

Esto permite aplicar operaciones de fecha que tengan en cuenta la zona horaria.

```
- `<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.
```

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el $hour operador para extraer el componente de hora de un campo de fecha y agrupar los datos en consecuencia.

Cree documentos de muestra

```
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") }
]);
```

Ejemplo de consulta

```
db.events.aggregate([
  {
    $project: {
      hour: { $hour: "$timestamp" }
    }
  },
  {
    $group: {
      _id: "$hour",
      count: { $sum: 1 }
    }
  },
  {
    $sort: { _id: 1 }
  }
]);
```

Salida

```
[
  { "_id": 8, "count": 1 },
  { "_id": 10, "count": 1 },
  { "_id": 12, "count": 1 },
  { "_id": 16, "count": 1 },
  { "_id": 23, "count": 1 }
]
```

Esta consulta agrupa los eventos por el componente horario del `timestamp` campo y cuenta el número de eventos de cada hora.

Ejemplos de código

Para ver un ejemplo de código para usar el `$hour` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$ifNull` operador se utiliza para devolver un valor especificado si la expresión de entrada se evalúa como nula o indefinida. Este operador puede resultar útil en situaciones en las que desee proporcionar un valor predeterminado o gestionar null/undefined casos.

Parámetros

- `expression`: La expresión que se va a evaluar.
- `replacement`: El valor que se devolverá si la <`expression`> evaluación es nula o indefinida.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$ifNull` operador para proporcionar un valor predeterminado cuando el `name` campo es nulo o no está definido.

Cree documentos de muestra

```
db.users.insertMany([
  { _id: 1, name: "John" },
  { _id: 2, name: null },
  { _id: 3 }
]);
```

Ejemplo de consulta

```
db.users.aggregate([
  {
    $project: {
      _id: 1,
      name: { $ifNull: ["$name", "No Name"] }
    }
  }
]);
```

Salida

```
[
  { "_id": 1, "name": "John" },
  { "_id": 2, "name": "No Name" },
  { "_id": 3, "name": "No Name" }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$ifNull` comando, elija la pestaña del idioma que desee usar:

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

El operador de `$in` agregación comprueba si existe un valor específico dentro de una matriz. Devuelve `true` si el valor se encuentra en la matriz o si `false` no.

Parámetros

- `value`: el valor que se va a buscar.
- `array`: La matriz en la que se va a buscar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$in` operador para comprobar si existe una habilidad específica en el conjunto de habilidades de cada empleado.

Cree documentos de muestra

```
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"] }
]);
```

Ejemplo de consulta

```
db.employees.aggregate([
  {
    $project: {
      name: 1,
      hasPython: { $in: ["Python", "$skills"] }
    }
  }
]);
```

Salida

```
[
  { _id: 1, name: 'Sarah', hasPython: true },
  { _id: 2, name: 'Mike', hasPython: false },
  { _id: 3, name: 'Emma', hasPython: true }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el operador de `$in` agregación, elija la pestaña correspondiente al idioma que desee usar:

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()
```

$índice OfArray

El `$indexOfArray` operador de Amazon DocumentDB se utiliza para buscar el índice de la primera aparición de un elemento especificado en una matriz. Este operador devuelve la posición de índice

basada en cero del primer elemento de la matriz que coincide con el valor especificado. Si no se encuentra el valor, devuelve -1.

Parámetros

- `array`: La matriz que se va a buscar.
- `value`: El valor que se va a buscar en la matriz.
- `start`: (opcional) La posición de la matriz desde la que iniciar la búsqueda. El valor predeterminado es 0.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra cómo utilizar el OfArray operador $index para buscar el índice de la primera aparición del elemento «mango» en la matriz «fruits» de cada documento.

Cree documentos de muestra

```
db.collection.insertMany([
  { _id: 1, fruits: ["apple", "banana", "cherry", "durian"] },
  { _id: 2, fruits: ["mango", "orange", "pineapple"] },
  { _id: 3, fruits: ["kiwi", "lemon", "mango"] }
]);
```

Ejemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
      _id: 1,
      fruitIndex: { $indexOfArray: ["$fruits", "mango"] }
    }
  }
]);
```

Salida

```
{ "_id" : 1, "fruitIndex" : 1 }
{ "_id" : 2, "fruitIndex" : 0 }
{ "_id" : 3, "fruitIndex" : 2 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$indexOfArray` comando, elija la pestaña del idioma que desee usar:

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()
```

$índice OfBytes

El OfBytes operador $index de Amazon DocumentDB se utiliza para buscar el índice inicial de una subcadena dentro de otra cadena, en función de las posiciones de los bytes de los caracteres. Esto puede resultar útil cuando se trabaja con datos de texto que pueden contener caracteres de varios bytes, como los que se encuentran en alfabetos no latinos.

Parámetros

- `string`: la cadena de entrada que se va a buscar.
- `substring`: La subcadena que se va a buscar en la cadena de entrada.
- `[<start>]`: (opcional) La posición inicial (basada en cero) de la búsqueda. Si no se especifica, la búsqueda comienza al principio de la cadena.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo se muestra el uso de `$indexOfBytes` para buscar el índice del primer carácter de guión en un conjunto de cadenas que representan la ubicación de los escritorios.

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.people.aggregate([
  { $project: { stateLocation: { $indexOfBytes: ["$Desk", "-"] } } }
]);
```

Salida

```
{ "_id" : 1, "stateLocation" : 11 }
{ "_id" : 2, "stateLocation" : 6 }
{ "_id" : 3, "stateLocation" : 7 }
{ "_id" : 4, "stateLocation" : 8 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$indexOfBytes` comando, elija la pestaña del idioma que desee usar:

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

El `$indexOfCP` operador de Amazon DocumentDB se utiliza para buscar el índice, en puntos de código (CP), de la primera aparición de una subcadena especificada dentro de una expresión de cadena. Esto puede resultar útil al analizar y extraer contenido de campos de cadenas.

Parámetros

- `string expression`: La cadena que se va a buscar.
- `substring`: La subcadena que se va a buscar.
- `[<start>]`: (opcional) La posición en la que se inicia la búsqueda (índice de base cero). El valor predeterminado es 0.

Ejemplo (MongoDB Shell)

En este ejemplo, utilizamos el `$indexOfCP` operador para buscar el índice de la primera aparición del carácter de guión, en el campo Escritorio de cada documento.

Cree documentos de muestra

```
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"}
])
```

Ejemplo de consulta

```
db.people.aggregate([
  { $project: { stateLocation: { $indexOfCP: [ "$Desk", "-"] } } }
])
```

Salida

```
{ "_id" : 1, "stateLocation" : 10 }
{ "_id" : 2, "stateLocation" : 6 }
{ "_id" : 3, "stateLocation" : 7 }
{ "_id" : 4, "stateLocation" : 8 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$indexOfCP` comando, elija la pestaña del idioma que desee usar:

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 etapa de `$indexStats` agregación de Amazon DocumentDB proporciona información sobre el uso de los índices dentro de una colección. Este operador le permite monitorear los patrones de acceso de sus índices, lo que puede ayudarlo a tomar decisiones informadas sobre la administración y la optimización de los índices.

Parámetros

Ninguno

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$indexStats` operador para analizar el uso del índice en una colección de Amazon DocumentDB.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.grocery.aggregate([
  { $indexStats: {} }
]);
```

Salida

```
[
  {
    "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")
    }
  }
]
```

En este ejemplo, el `$indexStats` operador muestra que se ha accedido al `_id_` índice 5 veces y que se ha accedido al `product_1` índice 10 veces desde el último restablecimiento o reinicio del servidor.

Ejemplos de código

Para ver un ejemplo de código para usar el `$indexStats` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$isArray` operador de Amazon DocumentDB se utiliza para comprobar si un campo de un documento es una matriz. Este operador puede resultar útil en canalizaciones de agregación y expresiones condicionales para gestionar campos de tipo matriz.

Parámetros

- `field`: La ruta del campo para comprobar si es una matriz.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo utilizar el `$isArray` operador para identificar documentos en los que el campo «inventario» es una matriz.

Cree documentos de muestra

```
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}]}
]);
```

Ejemplo de consulta

```
db.videos.aggregate([
  {
    $match: {
      $isArray: "$inventory"
    }
  },
  {
    $project: {
      _id: 1,
      name: 1,
      "inventory.location": 1,
      "inventory.count": 1
    }
  }
```

```
]).pretty();
```

Salida

```
{
  "_id": 4,
  "name": "Bravemind",
  "inventory": [
    {
      "location": "Mason City",
      "count": 250
    },
    {
      "location": "Des Moines",
      "count": 1000
    },
    {
      "location": "Ames",
      "count": 500
    }
  ]
}
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$isArray` comando, elija la pestaña del idioma que desee usar:

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()
```

$es un número

Nuevo desde la versión 8.0.1.

El `$isNumber` operador de Amazon DocumentDB devuelve un booleano que indica si una expresión determinada se resuelve en un tipo numérico (entero, decimal, doble, largo). Tenga en cuenta que las matrices no se consideran numéricas, incluso si todos los elementos de la matriz son números. Por ejemplo, si `$isNumber` se aplica a los retornos de valores`[1, 2, 3]`. `false`

Parámetros

- `expression`: cualquier expresión para comprobar si se resuelve en un tipo numérico.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$isNumber` operador para comprobar si los valores de los campos son numéricos.

Cree documentos de muestra

```
db.data.insertMany([
  {_id: 1, value: 42},
  {_id: 2, value: "hello"},
  {_id: 3, value: 3.14},
  {_id: 4, value: null}
]);
```

Ejemplo de consulta

```
db.data.aggregate([
  { $project: { isNum: { $isNumber: "$value" } } }
]);
```

Salida

```
[
  {_id: 1, isNum: true},
  {_id: 2, isNum: false},
  {_id: 3, isNum: true},
  {_id: 4, isNum: false}
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$isNumber` operador, elige la pestaña del idioma que quieres usar:

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

El `$isoDayOfWeek` operador de Amazon DocumentDB devuelve el día ISO de la semana de una fecha en forma de valor entero. El sistema ISO de fechas semanales define cada semana a partir de un lunes y termina un domingo, siendo la semana 1 la semana que contiene el primer jueves del año.

Parámetros

- `expression`: la expresión de fecha para la que se devuelve el día ISO de la semana.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$isoDayOfWeek` operador para recuperar el día ISO de la semana de un conjunto de documentos de eventos.

Cree documentos de muestra

```
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") }
]);
```

Ejemplo de consulta

```
db.events.aggregate([
  { $project: {
    _id: 1,
    eventDate: 1,
    isoDayOfWeek: { $isoDayOfWeek: "$eventDate" }
  }}
]);
```

Salida

```
[
  { "_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 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$isoDayOfWeek` comando, elija la pestaña del idioma que desee usar:

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

El `$isoWeek` operador de Amazon DocumentDB devuelve el número de semana ISO de una fecha. El sistema ISO de fechas semanales es una forma de numerar las semanas de un año, en el que la primera semana de un nuevo año es la semana que contiene el primer jueves de ese año. Esto es

diferente del calendario gregoriano, donde la primera semana de un nuevo año es la semana que contiene el 1 de enero.

Parámetros

Ninguno

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$isoWeek` operador para recuperar el número de semana ISO de una fecha determinada.

Cree documentos de muestra

```
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") }
])
```

Ejemplo de consulta

```
db.dates.aggregate([
  {
    $project: {
      _id: 1,
      isoWeek: { $isoWeek: "$date" }
    }
  }
])
```

Salida

```
[
  { "_id": 1, "isoWeek": 52 },
  { "_id": 2, "isoWeek": 52 },
  { "_id": 3, "isoWeek": 1 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$isoWeek` comando, elija la pestaña del idioma que desee usar:

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

El `$isoWeekYear` operador de Amazon DocumentDB devuelve el número del año semanal ISO 8601 de una fecha determinada. El número del año semanal ISO se diferencia del año civil gregoriano en que el año semanal puede ser diferente del año natural, especialmente al principio y al final del año.

Parámetros

- `expression`: la expresión de fecha para la que se devuelve el número del año semanal ISO 8601.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo utilizar el `$isoWeekYear` operador para recuperar el año semanal ISO 8601 para el campo de fecha de cada documento de la colección de eventos.

Cree documentos de muestra

```
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") }
]);
```

Ejemplo de consulta

```
db.events.aggregate([
  { $project: {
    name: 1,
    isoWeekYear: { $isoWeekYear: "$date" }
  }}
```

```
]);
```

Salida

```
[
  { "_id" : 1, "name" : "Event 1", "isoWeekYear" : 2023 },
  { "_id" : 2, "name" : "Event 2", "isoWeekYear" : 2023 },
  { "_id" : 3, "name" : "Event 3", "isoWeekYear" : 2023 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$isoWeekYear` comando, elija la pestaña del idioma que desee usar:

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

El `$last` operador de Amazon DocumentDB se usa para devolver el último elemento de una matriz que coincide con los criterios de la consulta. Resulta especialmente útil para recuperar el elemento más reciente o el último de una matriz que cumple una condición específica.

Parámetros

- `expression`: la expresión que coincide con los elementos de la matriz.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$last` operador en combinación con `$filter` la recuperación del último elemento de una matriz que cumple una condición específica (p. ej., el tema es «ciencia»).

Cree documentos de muestra

```
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 }
    ]
  }
]);
```

Ejemplo de consulta

```
db.collection.aggregate([
    { $match: { name: "John" } },
    {
      $project: {
        name: 1,
        lastScienceScore: {
          $last: {
            $filter: {
              input: "$scores",
              as: "score",
              cond: { $eq: ["$$score.subject", "science"] }
            }
          }
        }
```

```
      }
    }
  ]);
```

Salida

```
[
  {
    _id: 1,
    name: 'John',
    lastScienceScore: { subject: 'science', score: 90 }
  }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$last` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve el valor del `expression` del último documento de cada ventana. Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

En el ejemplo siguiente se dividen los documentos`sticker`, se ordenan `hour` las particiones y se devuelve el último precio registrado en cada partición.

```
db.stockPrices.aggregate([
  {
```

```
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        lastPrice: {
          $last: "$price",
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Cada documento se complementa con `lastPrice` el precio del último documento de su ventana (el último `hour` de la partición).

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$last` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran tanto el uso de expresiones (in`$project`) como el uso de operadores de ventanas (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()
```

$Último N

Nuevo a partir de la versión 8.0.1.

El `$lastN` operador de Amazon DocumentDB devuelve los últimos N elementos. Cuando se usa como acumulador en una `$group` etapa, devuelve una matriz con los últimos N valores de cada grupo. Cuando se usa como operador de expresión matricial, devuelve los últimos N elementos de una matriz.

Parámetros

- `input`: la expresión que se convierte en el campo o la matriz desde los que se devuelven los valores.
- n: un entero positivo que especifica cuántos valores se devolverán. Cuando se usa como acumulador en una `$group` etapa, también n puede ser una expresión, siempre que se resuelva como un entero positivo según el campo del grupo`_id`.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$lastN` acumulador para recuperar las dos últimas cantidades de cada artículo durante la agregación.

> **Note**
>
> `$lastN`selecciona los valores en el orden en que los documentos llegan al `$group` escenario. Para devolver los últimos N valores de un orden específico (por ejemplo, por fecha o puntuación), añada una `$sort` etapa antes`$group`.

Crea documentos de muestra

```
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") }
]);
```

Ejemplo de consulta

```
db.sales.aggregate([
  { $group: { _id: "$item", lastTwoQuantities: { $lastN: { input: "$quantity", n:
 2 } } } }
]);
```

Salida

```
[
  { "_id": "abc", "lastTwoQuantities": [5, 8] },
  { "_id": "xyz", "lastTwoQuantities": [7, 3] }
]
```

Ejemplo de uso de expresiones (MongoDB Shell)

El `$lastN` operador también se puede usar como expresión dentro de una `$project` etapa para devolver los últimos N elementos de un campo matricial.

Crea documentos de muestra

```
db.inventory.insertMany([
  { _id: 1, item: "abc", tags: ["red", "green", "blue", "yellow", "purple"] },
  { _id: 2, item: "xyz", tags: ["alpha", "beta", "gamma"] }
]);
```

Ejemplo de consulta

```
db.inventory.aggregate([
  { $project: {
      lastThreeTags: { $lastN: { input: "$tags", n: 3 } }
    }}
]);
```

Salida

```
[
```

```
  { "_id": 1, "lastThreeTags": ["blue", "yellow", "purple"] },
  { "_id": 2, "lastThreeTags": ["alpha", "beta", "gamma"] }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$lastN` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve una matriz con hasta los últimos n valores de los documentos de cada ventana. Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

> **Note**
>
> Cuando se usa como operador de ventana en`$setWindowFields`, `$lastN` está limitado a 100 MB de datos intermedios. Una operación que supera este límite devuelve un error.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

En el ejemplo siguiente se dividen los documentos`ticker`, se ordenan las `hour` particiones y se devuelven los dos precios más recientes vistos desde el principio de la partición hasta el documento actual.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
```

```
      sortBy: { hour: 1 },
      output: {
        lastTwoSoFar: {
          $lastN: { input: "$price", n: 2 },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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] }
]
```

Cada documento se complementa con `lastTwoSoFar` los dos últimos `price` valores de su ventana (en `hour` orden).

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$lastN` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran el uso del acumulador (in`$group`), el uso de expresiones (in`$project`) y el uso del operador de ventana (in`$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

El `$let` operador de Amazon DocumentDB se usa para vincular variables a valores y usar esas variables en la expresión. Le permite definir variables locales que se pueden usar en expresiones posteriores dentro de la misma etapa del proceso de agregación.

Parámetros

- `vars`: un objeto que define las variables que se van a utilizar en la expresión.
- `in`: La expresión en la que se utilizan las variables definidas en el parámetro vars.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra el uso del `$let` operador para calcular el área de un rectángulo.

Cree documentos de muestra

```
db.shapes.insertMany([
  { name: "Rectangle 1", length: 5, width: 3 },
  { name: "Rectangle 2", length: 7, width: 4 },
  { name: "Rectangle 3", length: 6, width: 2 }
]);
```

Ejemplo de consulta

```
db.shapes.aggregate([
  {
    $project: {
      name: 1,
      area: {
        $let: {
          vars: {
```

```
            length: "$length",
            width: "$width"
          },
          in: {
            $multiply: ["$$length", "$$width"]
          }
        }
      }
    }
  }
])
```

Salida

```
[
  {
    "_id": ObjectId("6161e5b1a3eba3c7f2960d03"),
    "name": "Rectangle 1",
    "area": 15
  },
  {
    "_id": ObjectId("6161e5b1a3eba3c7f2960d04"),
    "name": "Rectangle 2",
    "area": 28
  },
  {
    "_id": ObjectId("6161e5b1a3eba3c7f2960d05"),
    "name": "Rectangle 3",
    "area": 12
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$let` comando, elija la pestaña del idioma que desee usar:

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

El `$limit` operador de Amazon DocumentDB se utiliza para restringir el número de documentos devueltos por una consulta. Es similar al operador `$limit` MongoDB, pero hay que tener en cuenta algunas consideraciones específicas al usarlo con Amazon DocumentDB.

En Amazon DocumentDB, el `$limit` operador es útil para la paginación, en la que se desea recuperar un subconjunto del total de documentos coincidentes. Le permite controlar la cantidad de documentos devueltos en cada respuesta, lo que mejora el rendimiento y reduce la cantidad de datos que se transfieren a través de la red.

Parámetros

- `limit`: El número máximo de documentos que se van a devolver. Debe ser un valor entero no negativo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$limit` operador para devolver un máximo de un documento que coincida con el filtro dado.

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.test.createIndex({ comments: "text" });

db.test.find({
  $and: [
    { star_rating: 5 },
    { $text: { $search: "couch" } }
  ]
}).limit(1);
```

Salida

```
[ { _id: 2, star_rating: 5, comments: 'comfortable couch' } ]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$limit` comando, elija la pestaña del idioma que desee usar:

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()
```

$lista SearchIndexes

Nuevo a partir de la versión 8.0.1.

La fase de `$listSearchIndexes` agregación de Amazon DocumentDB devuelve información sobre los índices de búsqueda existentes en una colección. Debe ser la primera etapa de un proceso de agregación.

Parámetros

- `name`: (opcional) El nombre del índice de búsqueda sobre el que se devolverá la información. Si se omite, se devuelven todos los índices de búsqueda de la colección.

Campos de salida

Cada documento devuelto contiene los siguientes campos:

- `name`: el nombre del índice de búsqueda.
- `status`: El estado de creación del índice. Uno de `READY` (creado y válido), `BUILDING` (hay una compilación simultánea en curso) o `FAILED` (la creación del índice no se completó correctamente).
- `queryable`: un booleano que indica si el índice se puede usar actualmente para atender consultas. Esto es `false` para los índices ocultos (que permanecen`READY`) y para los índices. `FAILED` Un índice oculto es el caso en el que `status` es `READY` pero `queryable` es. `false`
- `latestDefinitionVersion`: un documento que contiene `version` (la versión en formato de índice) y `createdAt` (la hora en que se creó el índice).

Sintaxis

```
db.collection.aggregate([
  { $listSearchIndexes: {} }
])

// Or filter by name:
db.collection.aggregate([
  { $listSearchIndexes: { name: "mySearchIndex" } }
])
```

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$listSearchIndexes` escenario para mostrar todos los índices de búsqueda de una colección.

Ejemplo de consulta

```
db.movies.aggregate([
  { $listSearchIndexes: {} }
]);
```

Salida

```
[
  {
    "name": "default",
    "status": "READY",
    "queryable": true,
    "latestDefinitionVersion": {
```

```
      "version": 2,
      "createdAt": ISODate("2026-05-11T21:12:57.974Z")
    }
  }
]
```

Un índice oculto permanece, `READY` pero informa`queryable: false`, porque no se puede usar para atender consultas mientras está oculto:

```
[
  {
    "name": "myHiddenIndex",
    "status": "READY",
    "queryable": false,
    "latestDefinitionVersion": {
      "version": 2,
      "createdAt": ISODate("2026-05-11T21:12:57.974Z")
    }
  }
]
```

Para filtrar por un nombre de índice específico:

```
db.movies.aggregate([
  { $listSearchIndexes: { name: "default" } }
]);
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$listSearchIndexes` escenario, elige la pestaña correspondiente al idioma que quieres usar:

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

El `$literal` operador de Amazon DocumentDB se utiliza para representar un valor literal dentro de una etapa de canalización de agregación. Le permite incluir un valor específico, como un número, una cadena o un booleano, sin interpretarlo como una referencia o expresión de campo.

Este operador resulta especialmente útil cuando se necesita incluir un valor literal como parte de un proceso de agregación más complejo, como cuando se crean filtros de consultas dinámicos o se realizan cálculos.

Parámetros

Ninguno

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$literal` operador para incluir un valor literal en una canalización de agregación. El `$literal` operador se utiliza para incluir el valor 18 como un valor literal en la expresión $gt. Esto permite que el proceso de agregación compare el campo de edad con el valor literal 18 para determinar si la persona es adulta.

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
      name: 1,
      age: 1,
      city: 1,
      isAdult: { $gt: ["$age", { $literal: 18 }] }
    }
  }
]);
```

Salida

```
[
  {
    "_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
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$literal` comando, elija la pestaña del idioma que desee usar:

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

Nuevo desde la versión 4.0.

El `$ln` operador de Amazon DocumentDB calcula el logaritmo natural (base e) de un número especificado. Devuelve el logaritmo del número en la base e.

Parámetros

- `expression`: El número para el que se calculará el logaritmo natural.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$log` operador para calcular el logaritmo natural de un número.

Cree documentos de muestra

```
db.numbers.insertMany([
  { _id: 1, value: 10 },
  { _id: 2, value: 100 },
  { _id: 3, value: 1000 }
]);
```

Ejemplo de consulta

```
db.numbers.aggregate([
  { $project: {
```

```
    _id: 1,
    naturalLog: { $ln: "$value" }
  }}
]);
```

Salida

```
[
  { "_id" : 1, "naturalLog" : 2.302585092994046 },
  { "_id" : 2, "naturalLog" : 4.605170185988092 },
  { "_id" : 3, "naturalLog" : 6.907755278982137 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$ln` comando, elija la pestaña del idioma que desee usar:

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

Nuevo desde la versión 4.0.

El `$log` operador de Amazon DocumentDB calcula el logaritmo natural de un número. Devuelve el logaritmo en base e del número especificado.

Parámetros

- `expression`: el número para el que se calculará el logaritmo natural.
- `base`: valor base para calcular el logaritmo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$log` operador para calcular el logaritmo natural de un número.

Cree documentos de muestra

```
db.numbers.insertMany([
  { _id: 1, value: 10 },
  { _id: 2, value: 100 },
  { _id: 3, value: 1000 }
]);
```

Ejemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    naturalLog: { $log: ["$value", 10] }
  }}
]);
```

Salida

```
[
  { "_id" : 1, "naturalLog" : 1 },
  { "_id" : 2, "naturalLog" : 2 },
  { "_id" : 3, "naturalLog" : 2.9999999999999996 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$log` comando, elija la pestaña del idioma que desee usar:

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

Nuevo desde la versión 4.0.

El `$log10` operador de Amazon DocumentDB se utiliza para calcular el logaritmo en base 10 de un número. Resulta útil para realizar cálculos logarítmicos en campos numéricos dentro de la canalización de agregación.

Parámetros

- `expression`: La expresión numérica para la que se va a calcular el logaritmo en base 10.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$log10` operador para calcular el logaritmo en base 10 de un campo numérico.

Cree documentos de muestra

```
db.numbers.insertMany([
  { _id: 1, value: 1 },
  { _id: 2, value: 10 },
  { _id: 3, value: 100 },
  { _id: 4, value: 1000 }
]);
```

Ejemplo de consulta

```
db.numbers.aggregate([
  {
    $project: {
      _id: 1,
      log10Value: { $log10: "$value" }
    }
  }
]);
```

Salida

```
[
  { "_id": 1, "log10Value": 0 },
  { "_id": 2, "log10Value": 1 },
  { "_id": 3, "log10Value": 2 },
  { "_id": 4, "log10Value": 3 }
```

```
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$log10` comando, elija la pestaña del idioma que desee usar:

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

El operador de `$lt` agregación compara dos valores y devuelve `true` si el primer valor es menor que el segundo; de lo contrario, devuelve`false`.

Parámetros

- `expression1`: El primer valor que se va a comparar.

- `expression2`: El segundo valor que se va a comparar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$lt` operador para identificar artículos con pocas existencias.

Cree documentos de muestra

```
db.warehouse.insertMany([
  { _id: 1, item: "Bolts", stock: 5 },
  { _id: 2, item: "Nuts", stock: 25 },
  { _id: 3, item: "Screws", stock: 8 }
]);
```

Ejemplo de consulta

```
db.warehouse.aggregate([
  {
    $project: {
      item: 1,
      stock: 1,
      lowStock: { $lt: ["$stock", 10] }
    }
  }
]);
```

Salida

```
[
  { _id: 1, item: 'Bolts', stock: 5, lowStock: true },
  { _id: 2, item: 'Nuts', stock: 25, lowStock: false },
  { _id: 3, item: 'Screws', stock: 8, lowStock: true }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el operador de `$lt` agregación, elija la pestaña correspondiente al idioma que desee usar:

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

El operador de `$lte` agregación compara dos valores y devuelve `true` si el primer valor es menor o igual que el segundo; de lo contrario, devuelve`false`.

Parámetros

- `expression1`: el primer valor que se va a comparar.
- `expression2`: El segundo valor que se va a comparar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$lte` operador para identificar artículos asequibles.

Cree documentos de muestra

```
db.menu.insertMany([
  { _id: 1, dish: "Salad", price: 8 },
  { _id: 2, dish: "Pasta", price: 12 },
  { _id: 3, dish: "Soup", price: 6 }
]);
```

Ejemplo de consulta

```
db.menu.aggregate([
  {
    $project: {
      dish: 1,
      price: 1,
      affordable: { $lte: ["$price", 10] }
```

```
    }
  }
]);
```

Salida

```
[
  { _id: 1, dish: 'Salad', price: 8, affordable: true },
  { _id: 2, dish: 'Pasta', price: 12, affordable: false },
  { _id: 3, dish: 'Soup', price: 6, affordable: true }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el operador de `$lte` agregación, elija la pestaña correspondiente al idioma que desee usar:

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 de `$lookup` agregación de Amazon DocumentDB le permite realizar una unión externa izquierda entre dos colecciones. Esta operación le permite combinar datos de varias colecciones en función de los valores de campo coincidentes. Resulta especialmente útil cuando necesita incorporar datos de colecciones relacionadas en los resultados de la consulta.

Parámetros

- `from`: el nombre de la colección con la que se realizará la unión.
- `localField`: El campo de los documentos de entrada con el que se compararáforeignField.
- `foreignField`: El campo de los documentos de la `from` colección con el que se compararálocalField.

- `as`: el nombre del nuevo campo que se va a añadir a los documentos de salida que contienen los documentos coincidentes de la `from` colección.
- `pipeline`: Opcional si se `foreignField` especifican `localField` y se especifican. Una canalización de agregación para ejecutarse en la `from` colección. La canalización devuelve los documentos de la `from` colección. Para devolver todos los documentos, especifique una canalización vacía`[]`. La canalización no puede incluir las `$merge` etapas `$out` o. Consulte Condiciones de unión y subconsultas correlacionadas (MongoDB Shell).
- `let`: opcional. Un documento que define las variables que se van a utilizar en las `pipeline` etapas. Utilice las expresiones variables para acceder a los campos desde los documentos de entrada que están unidos al`pipeline`. Para hacer referencia a una variable en las etapas del proceso, utilice la `"$$<variable>"` sintaxis. Consulte Condiciones de unión y subconsultas correlacionadas (MongoDB Shell).

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra una `$lookup` operación sencilla que une los datos de la `orders` colección con la `customers` colección.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.customers.aggregate([
  {
    $lookup: {
      from: "orders",
      localField: "_id",
      foreignField: "customer_id",
```

```
      as: "orders"
    }
  }
]);
```

Salida

```
[
  {
    _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 } ]
  }
]
```

Condiciones de unión y subconsultas correlacionadas (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

Además de una única igualación, `$lookup` en Amazon DocumentDB se puede ejecutar una canalización de agregación en la colección unida. Esto admite subconsultas no correlacionadas, que ejecutan la misma subconsulta para cada documento de entrada, y subconsultas correlacionadas, que utilizan variables definidas en los campos de cada documento de entrada para obtener resultados correlacionados con ese documento.

Sintaxis

Para ejecutar una subconsulta correlacionada, defina las variables de los campos del documento de entrada con y haga referencia a ellas en las etapas siguientes: `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>
  }
}
```

Parámetros

- `let`: un documento que define las variables a partir de los campos de los documentos de entrada. Para hacer referencia a una variable en las etapas del proceso, utilice la `"$$<variable>"` sintaxis. Se puede acceder a las variables desde las etapas de`pipeline`, incluidas `$lookup` las etapas adicionales anidadas en. `pipeline` Una `$match` etapa requiere el uso del `$expr` operador para acceder a las variables. Omita `let` ejecutar una subconsulta no correlacionada.
- `pipeline`: la canalización de agregación que se ejecutará en la colección. `from` La canalización no puede acceder directamente a los campos de los documentos de entrada; en su lugar, defina las variables con ellas `let` y haga referencia a ellas. Para devolver todos los documentos, especifique una canalización vacía`[]`.

Los operadores `$eq`,`$gt`,`$gte`,`$lt`, y de `$lte` comparación colocados en un `$expr` operador dentro de una `$match` etapa pueden usar un índice en la `from` colección cuando comparan un campo de la colección combinada con una `let` variable o una constante.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

El siguiente ejemplo une a cada cliente con sus pedidos y conserva solo los pedidos cuyo valor `total` es superior o igual al `min_total` valor propio de ese cliente. Tanto la condición de unión como el filtro por cliente hacen referencia a `let` las siguientes variables: `$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"
    }
  }
]);
```

Salida

```
[
  { _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: [] }
]
```

Para cada cliente, la subconsulta se evalúa con los `min_total` valores `_id` y de ese cliente, por lo que cada documento recibe su propio resultado correlacionado.

Subconsultas correlacionadas mediante una sintaxis concisa (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

Amazon DocumentDB también admite la sintaxis concisa de subconsultas correlacionadas, que combina la coincidencia de igualdad en y con un. `localField` `foreignField` `pipeline` La sintaxis concisa elimina el requisito de expresar la coincidencia de igualdad dentro de un operador en una `$expr` etapa. `$match`

Sintaxis

```
{
  $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>
  }
}
```

Ejemplo de consulta

El siguiente ejemplo devuelve los mismos resultados que el ejemplo anterior de subconsulta correlacionada. La coincidencia de igualdad entre `_id` y `customer_id` se expresa con `localField` y `foreignField` en lugar de una `$expr` condición:

```
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"
    }
  }
]);
```

Salida

```
[
  { _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: [] }
]
```

Etapas compatibles en las subconsultas correlacionadas

Cuando se utiliza una `$lookup` canalización`let`, las siguientes etapas pueden hacer referencia a las `"$$<variable>"` variables:

- `$match`(usando el `$expr` operador)
- `$project`
- `$addFields`
- `$set`
- `$group`
- `$bucket`
- `$bucketAuto`
- `$sortByCount`
- `$lookup` (anidado)
- `$graphLookup`

Las etapas que no evalúan expresiones variables, como,`$sort`,, y `$skip``$limit`,`$count`,,`$sample`,`$unwind`,,`$unset`, también se pueden utilizar en el proceso, pero no pueden hacer referencia a las variables.

Las siguientes etapas no se admiten en una `$lookup` canalización que utilice`let`: `$facet` `$redact``$replaceRoot`,`$replaceWith`,`$geoNear`,`$setWindowFields`,`$fill`, y `$match` con`$text`.

Restricciones

- `pipeline`No pueden incluir las `$merge` etapas `$out` o.

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$lookup` comando, elija la pestaña correspondiente al idioma que desea usar:

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()
```

Ejemplos de códigos de subconsultas correlacionados

Para ver un ejemplo de código sobre cómo usar el `$lookup` escenario con una subconsulta correlacionada (usando `let` y`pipeline`), elige la pestaña del idioma que quieres usar:

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

Nuevo desde la versión 4.0.

No es compatible con el clúster de Elastic.

El `$ltrim` operador de Amazon DocumentDB se utiliza para eliminar los caracteres principales de una cadena. De forma predeterminada, elimina los espacios en blanco iniciales, pero también puede especificar un conjunto de caracteres para eliminarlos pasando el argumento chars.

Parámetros

- `input`: la cadena de entrada de la que se van a eliminar los espacios en blanco iniciales.
- `chars`: (opcional) Para eliminar caracteres específicos.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso de `$ltrim` para eliminar caracteres específicos («*») del principio de una cadena.

Cree documentos de muestra

```
db.collection.insertMany([
  { name: " *John Doe", age: 30 },
  { name: "Jane Doe*", age: 25 },
  { name: "  Bob Smith  ", age: 35 }
]);
```

Ejemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
      _id: 0,
      name: {
        $ltrim: { input: "$name", chars: " *" }
      },
      age: 1
    }
  }
]);
```

Salida

```
[
  { "name": "John Doe", "age": 30 },
  { "name": "Jane Doe ", "age": 25 },
```

```
  { "name": "Bob Smith  ", "age": 35 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$ltrim` comando, elija la pestaña del idioma que desee usar:

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

El $map operador de Amazon DocumentDB le permite aplicar una expresión específica a cada elemento de una matriz y devolver una nueva matriz con los elementos transformados. Este operador es especialmente útil para manipular y transformar los datos de las matrices, lo que puede ayudar a simplificar el código de la aplicación y mejorar el rendimiento de las consultas al llevar el procesamiento de las matrices al nivel de la base de datos.

Parámetros

- `input`: La matriz que se va a transformar.
- `as`: (opcional) El nombre de la variable que se utilizará en la expresión in para representar el elemento actual que se está procesando.
- `in`: la expresión que se va a aplicar a cada elemento de la matriz de entrada.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el operador $map para transformar una matriz de números, duplicando cada valor.

Cree documentos de muestra

```
db.collection.insertMany([
  { _id: 1, numbers: [1, 2, 3, 4, 5] },
  { _id: 2, numbers: [10, 20, 30, 40, 50] }
])
```

Ejemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
      doubledNumbers: { $map: { input: "$numbers", as: "num", in: { $multiply: ["$
$num", 2] } } }
```

```
    }
  }
])
```

Salida

```
[
  { _id: 1, doubledNumbers: [2, 4, 6, 8, 10] },
  { _id: 2, doubledNumbers: [20, 40, 60, 80, 100] }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el $map comando, elija la pestaña del idioma que desee usar:

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 etapa de `$match` canalización de Amazon DocumentDB se utiliza para filtrar los documentos de entrada y ver solo aquellos que coincidan con los criterios de consulta especificados. Es una de las etapas de canalización más utilizadas en las operaciones de agregación. La `$match` etapa se aplica antes que cualquier otra fase de la canalización, lo que permite reducir de manera eficiente la cantidad de documentos que deben procesarse en las etapas posteriores.

Parámetros

- `query`: documento que expresa los criterios de selección de la operación. El documento de consulta utiliza la misma sintaxis que el `find()` método.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso de la `$match` etapa para filtrar documentos en función de un valor de campo específico.

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.collection.aggregate([
  { $match: { age: { $gt: 30 } } },
  { $project: { _id: 1, name: 1, city: 1 } }
]);
```

Salida

```
[
  { "_id": 3, "name": "Bob", "city": "Chicago" },
  { "_id": 4, "name": "Alice", "city": "Miami" }
]
```

La `$match` etapa filtra los documentos para incluir solo aquellos en los que el `age` campo es superior a 30.

Ejemplos de código

Para ver un ejemplo de código para usar el `$match` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El operador de `$max` agregación devuelve el valor máximo de un campo especificado en un conjunto de documentos. Este operador es útil para buscar el valor más alto en un conjunto de documentos.

Parámetros

- `expression`: la expresión que se va a utilizar para calcular el valor máximo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$max` operador para encontrar la puntuación máxima en una colección de documentos de los estudiantes. La `$group` etapa agrupa todos los documentos y el `$max` operador se usa para calcular el valor máximo del `score` campo en todos los documentos.

Cree documentos de muestra

```
db.students.insertMany([
  { name: "John", score: 85 },
  { name: "Jane", score: 92 },
  { name: "Bob", score: 78 },
  { name: "Alice", score: 90 }
])
```

Ejemplo de consulta

```
db.students.aggregate([
  { $group: { _id: null, maxScore: { $max: "$score" } } },
  { $project: { _id: 0, maxScore: 1 } }
])
```

Salida

```
[ { maxScore: 92 } ]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$max` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve el valor máximo de los documentos de cada ventana.

Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

> **Note**
>
> Cuando se usa como operador de ventana en`$setWindowFields`, `$max` está limitado a 100 MB de datos intermedios. Una operación que supera este límite devuelve un error.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

En el ejemplo siguiente se dividen los documentos`ticker`, se ordena cada partición y se devuelve el precio más alto registrado desde el principio de la partición hasta el documento actual. `hour`

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        highestSoFar: {
          $max: "$price",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Cada documento se amplía con `highestSoFar` el máximo de valores `price` dentro de su partición, incluido el documento actual.

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$max` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran tanto el uso del acumulador (en`$group`) como el uso del operador de ventanas (en`$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()
```

$ máx. N

Nuevo a partir de la versión 8.0.1.

El `$maxN` operador de Amazon DocumentDB devuelve los N valores más altos. Cuando se usa como acumulador en una `$group` etapa, devuelve una matriz con los N valores máximos de cada grupo. Cuando se usa como operador de expresión matricial, devuelve los N elementos máximos de una matriz.

Parámetros

- `input`: una expresión que se convierte en la matriz o el campo desde el que se devuelven los `n` valores máximos.
- `n`: expresión que se resuelve en un entero positivo que especifica cuántos valores máximos se devolverán.

Comportamiento

Comportamiento del operador de expresiones matriciales

- No puede especificar un valor `n` inferior a`1`.
- `$maxN`filtra `null` los valores encontrados en la `input` matriz.
- Si el valor especificado `n` es mayor o igual al número de elementos de la `input` matriz, `$maxN` devuelve todos los elementos de la `input` matriz.
- Si se `input` resuelve en un valor que no es de matriz, la operación de agregación produce errores.
- Si `input` contiene elementos numéricos y de cadena, los elementos de cadena se ordenan antes que los numéricos según el orden de comparación de BSON.

Comportamiento del acumulador

- Cuando se usa como acumulador, `n` debe ser una expresión integral positiva que sea una constante o que dependa del valor de. `_id` `$group`
- `$maxN`filtra los valores nulos y faltantes.
- Si el grupo contiene menos de `n` elementos, `$maxN` devuelve todos los elementos del grupo.

- `$maxN`compara los datos de entrada siguiendo el orden de comparación de BSON para determinar el tipo de salida apropiado. Cuando los datos de entrada contienen varios tipos de datos, el tipo `$maxN` de salida es el más alto en el orden de comparación.

Orden de salida

`$maxN`devuelve valores sin ningún orden de clasificación en particular. Si es obligatorio garantizar un orden de clasificación determinado, utilícelo `$topN` en su lugar o envuelva el resultado con`$sortArray`.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$maxN` acumulador para recuperar las dos puntuaciones más altas de cada asignatura.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.scores.aggregate([
  { $group: { _id: "$subject", highestTwo: { $maxN: { input: "$score", n: 2 } } } }
]);
```

Salida

```
[
  { "_id": "math", "highestTwo": [85, 93] },
  { "_id": "science", "highestTwo": [88, 90] }
]
```

Ejemplo de uso de expresiones (MongoDB Shell)

El `$maxN` operador también se puede usar como expresión dentro de una `$project` etapa para devolver los N elementos más grandes de un campo de matriz.

Crea documentos de muestra

```
db.readings.insertMany([
  { _id: 1, sensor: "A", values: [45, 12, 78, 3, 56] },
  { _id: 2, sensor: "B", values: [90, 23, 67, 11, 44] }
]);
```

Ejemplo de consulta

```
db.readings.aggregate([
  { $project: {
      highestThree: { $maxN: { input: "$values", n: 3 } }
    }}
]);
```

Salida

```
[
  { "_id": 1, "highestThree": [45, 56, 78] },
  { "_id": 2, "highestThree": [44, 90, 67] }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$maxN` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve una matriz con hasta los valores n más altos de los documentos de cada ventana. Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

> **Note**
>
> Cuando se usa como operador de ventana en`$setWindowFields`, `$maxN` está limitado a 100 MB de datos intermedios. Una operación que supera este límite devuelve un error.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

En el ejemplo siguiente se dividen los documentos`ticker`, se ordenan las `hour` particiones y se devuelven los dos precios más altos desde el principio de la partición hasta el documento actual.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        highestTwoSoFar: {
          $maxN: { input: "$price", n: 2 },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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] }
```

```
]
```

Cada documento se complementa con `highestTwoSoFar` los dos `price` valores más altos de su ventana. Como se indica en la sección Ordenación de los resultados, el orden de los valores dentro de la matriz devuelta no está garantizado.

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$maxN` del operador, elija la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran el uso del acumulador (in`$group`), el uso de expresiones (in`$project`) y el uso del operador de ventana (in`$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

Nuevo a partir de la versión 8.0.1.

El `$median` operador de Amazon DocumentDB calcula la mediana del valor de los datos numéricos. Como acumulador, calcula la mediana de los valores numéricos de los documentos de un grupo en fase de proceso de agregación. `$group` Como expresión, calcula la mediana de una matriz de números.

Parámetros

- `input`: expresión que se resuelve en un valor numérico o en una matriz de valores numéricos.
- `method`: cadena que especifica el método de cálculo. Actualmente solo `"approximate"` es compatible, que utiliza el algoritmo t-digest.

Comportamiento

El `"approximate"` método usa el algoritmo t-digest para calcular una mediana aproximada. El resultado es un valor existente del conjunto de datos en lugar de una interpolación entre valores. La precisión mejora a medida que aumenta la cantidad de puntos de datos.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$median` operador para calcular la puntuación media de una prueba por clase.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.students.aggregate([
  { $group: {
      _id: "$class",
      medianScore: { $median: { input: "$score", method: "approximate" } }
    }}
]);
```

Salida

```
[
  { "_id": "A", "medianScore": 85 },
  { "_id": "B", "medianScore": 80 }
]
```

Ejemplo de uso de expresiones (MongoDB Shell)

El `$median` operador también se puede usar como expresión dentro de una `$project` etapa para calcular la mediana de un campo de matriz.

Cree documentos de muestra

```
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] }
]);
```

Ejemplo de consulta

```
db.surveys.aggregate([
  { $project: {
      medianRating: { $median: { input: "$ratings", method: "approximate" } }
    }}
]);
```

Salida

```
[
  { "_id": 1, "medianRating": 5 },
  { "_id": 2, "medianRating": 30 },
  { "_id": 3, "medianRating": 2 }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$median` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, calcula la mediana de los valores numéricos de

los documentos de cada ventana. Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

> **Note**
>
> Cuando se usa como operador de ventana en`$setWindowFields`, `$median` está limitado a 100 MB de datos intermedios. Una operación que supera este límite devuelve un error.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

El siguiente ejemplo divide los documentos `city` y calcula la mediana de `reading` todos los documentos de cada partición.

```
db.temperatures.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$city",
      output: {
        medianReading: {
          $median: { input: "$reading", method: "approximate" },
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Cada documento se complementa con `medianReading` la mediana aproximada de `reading` todos los documentos de su partición.

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$median` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran el uso del acumulador (in`$group`), el uso de expresiones (in`$project`) y el uso del operador de ventana (in`$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

El operador de `$meta` agregación accede a los metadatos asociados a los documentos de una canalización de agregación. Se suele utilizar para recuperar las puntuaciones de las búsquedas de texto y ordenar los resultados por relevancia.

Parámetros

- `textScore`: recupera la puntuación de búsqueda de texto que indica la relevancia del documento para la consulta de búsqueda.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$meta` operador en una canalización de agregación para recuperar y ordenar las puntuaciones de búsqueda de texto.

Cree documentos de muestra

```
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." }
]);
```

Ejemplo de consulta

```
db.articles.aggregate([
```

```
  { $match: { $text: { $search: "Python" } } },
  { $addFields: { score: { $meta: "textScore" } } },
  { $sort: { score: -1 } }
]);
```

Salida

```
[
  {
    _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
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el operador de `$meta` agregación, elija la pestaña correspondiente al idioma que desee usar:

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()
```

$merge

Introducido en 8.0

La fase de `$merge` agregación de Amazon DocumentDB se utiliza para combinar los resultados de la fase de canalización anterior en una colección de destino. Esto resulta útil para actualizar o insertar documentos en una colección de destino en función de los datos de los documentos de entrada.

La `$merge` etapa permite realizar varias acciones en función de la condición de coincidencia entre los documentos de entrada y la colección de destino, como:

```
- Insert new documents
- Update existing documents
- Delete documents
- Fail the operation if there are any conflicts
```

Parámetros

- `into`: (obligatorio) El nombre de la colección de destino en la que se combinarán los documentos de entrada.
- on: (obligatorio) Los campos que se van a utilizar como condición de coincidencia entre los documentos de entrada y la colección de destino.
- `whenMatched`: (opcional) La acción que se debe realizar cuando el documento de entrada coincide con un documento existente en la colección de destino. Los valores admitidos son: `"merge"`, `"replace"`, `"keepExisting"` y `"fail"`. Amazon DocumentDB no admite la especificación de una canalización de agregación como `whenMatched` valor.
- `whenNotMatched`: (opcional) La acción que se debe realizar cuando el documento de entrada no coincide con ningún documento de la colección de destino. Los valores admitidos son: `"insert"` y`"fail"`.
- `bypassDocumentValidation`: (opcional) Si se establece en`true`, los documentos escritos en la colección de destino por `$merge` fase omiten cualquier regla de validación de esquemas (`$jsonSchema`) definida en esa colección. El valor predeterminado es`false`, que hace cumplir las reglas de validación de la colección de destino. Esto es compatible con Amazon DocumentDB 8.0.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar la `$merge` etapa para actualizar una `users` colección con datos nuevos de una canalización de entrada.

Cree documentos de muestra

```
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" }
```

```
]);
```

Ejemplo de consulta

```
db.inputData.aggregate([
  {
    $merge: {
      into: "users",
      on: "_id",
      whenMatched: "merge",
      whenNotMatched: "insert"
    }
  }
])
```

Salida

Tras ejecutar la `$merge` canalización, la `users` colección contendrá los siguientes documentos:

```
[
  { _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" }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$merge` comando, elige la pestaña del idioma que quieres usar:

Node.js

Este es un ejemplo del uso del operador $merge en una Node.js aplicación:

```
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

Este es un ejemplo del uso del operador $merge en una aplicación de Python:

```
from pymongo 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

El `$mergeObjects` operador de Amazon DocumentDB se utiliza para combinar varios documentos u objetos en un único documento. Este operador resulta especialmente útil cuando se necesita combinar el contenido de dos o más documentos u objetos, lo que podría sobrescribir los valores de un objeto con los de otro.

Parámetros

- `expression1`: el primer objeto que se va a fusionar.
- `expression2`: (opcional) El segundo objeto que se va a fusionar.
- `expression3`: (opcional) Objetos adicionales que se van a fusionar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$mergeObjects` operador para combinar dos objetos.

Cree documentos de muestra

```
db.collection.insertMany([
  { "_id": 1, "name": "John", "address": { "city": "New York", "state": "NY" } },
  { "_id": 2, "name": "Jane", "address": { "city": "Los Angeles", "state": "CA" } }
]);
```

Ejemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
      "combinedAddress": {
        $mergeObjects: ["$address", { "country": "USA" }]
      }
    }
  }
```

```
])
```

Salida

```
[
  {
    "_id": 1,
    "combinedAddress": {
      "city": "New York",
      "state": "NY",
      "country": "USA"
    }
  },
  {
    "_id": 2,
    "combinedAddress": {
      "city": "Los Angeles",
      "state": "CA",
      "country": "USA"
    }
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$mergeObjects` comando, elija la pestaña del idioma que desee usar:

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

El `$millisecond` operador de Amazon DocumentDB se utiliza para extraer la parte de milisegundos de un valor de fecha.

Parámetros

Ninguno

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo utilizar el `$millisecond` operador para extraer la parte de milisegundos de un valor de fecha.

Cree documentos de muestra

```
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")
  }
])
```

Ejemplo de consulta

```
db.events.aggregate([
  {
    $project: {
      name: 1,
      milliseconds: { $millisecond: "$timestamp" }
    }
  }
])
```

Salida

```
[
  {
    "_id": ObjectId("644332a42054ed1b0d15f0c1"),
```

```
    "name": "Event 1",
    "milliseconds": 123
  },
  {
    "_id": ObjectId("644332a42054ed1b0d15f0c2"),
    "name": "Event 2",
    "milliseconds": 456
  },
  {
    "_id": ObjectId("644332a42054ed1b0d15f0c3"),
    "name": "Event 3",
    "milliseconds": 789
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$millisecond` comando, elija la pestaña del idioma que desee usar:

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

El operador de `$min` agregación devuelve el valor mínimo de un campo especificado en un conjunto de documentos. Este operador es útil para buscar el valor más bajo en un conjunto de documentos.

Parámetros

- `expression`: La expresión que se va a evaluar. Puede ser una ruta de campo, una variable o cualquier expresión que dé como resultado un valor.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$min` operador para encontrar el valor mínimo del `age` campo en varios documentos.

Cree documentos de muestra

```
db.users.insertMany([
  { name: "John", age: 35 },
  { name: "Jane", age: 28 },
  { name: "Bob", age: 42 },
  { name: "Alice", age: 31 }
]);
```

Ejemplo de consulta

```
db.users.aggregate([
  { $group: { _id: null, minAge: { $min: "$age" } } },
  { $project: { _id: 0, minAge: 1 } }
])
```

Salida

```
[ { minAge: 28 } ]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$min` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve el valor mínimo de los documentos de cada ventana. Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

> **Note**
>
> Cuando se usa como operador de ventana en`$setWindowFields`, `$min` está limitado a 100 MB de datos intermedios. Una operación que supera este límite devuelve un error.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

En el ejemplo siguiente se dividen los documentos`ticker`, se ordena cada partición y se devuelve el precio más bajo registrado desde el principio de la partición hasta el documento actual. `hour`

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        lowestSoFar: {
          $min: "$price",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Cada documento se amplía con`lowestSoFar`, como mínimo, el documento actual incluido `price` dentro de su partición.

Ejemplos de código

Para ver un ejemplo de código sobre el uso $min del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran tanto el uso del acumulador (en$group) como el uso del operador de ventanas (en$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()
```

$ mN

Nuevo a partir de la versión 8.0.1.

El `$minN` operador de Amazon DocumentDB devuelve los N valores más pequeños. Cuando se usa como acumulador en una `$group` etapa, devuelve una matriz con los N valores mínimos de cada grupo. Cuando se usa como operador de expresión matricial, devuelve los N elementos mínimos de una matriz.

Parámetros

- `input`: una expresión que se convierte en la matriz o el campo desde el que se devuelven los n valores mínimos.

- n: expresión que se resuelve en un entero positivo que especifica el número de valores mínimos que se devolverán.

Comportamiento

Comportamiento del operador de expresiones matriciales

- No puede especificar un valor n inferior a `1`.
- `$minN`filtra `null` los valores encontrados en la `input` matriz.
- Si el valor especificado n es mayor o igual al número de elementos de la `input` matriz, `$minN` devuelve todos los elementos de la `input` matriz.
- Si se `input` resuelve en un valor que no es de matriz, la operación de agregación produce errores.
- Si `input` contiene elementos numéricos y de cadena, los elementos numéricos se ordenan antes que los elementos de cadena según el orden de comparación de BSON.

Comportamiento del acumulador

- Cuando se usa como acumulador, n debe ser una expresión integral positiva que sea una constante o que dependa del valor de. `_id` `$group`
- `$minN`filtra los valores nulos y faltantes.
- Si el grupo contiene menos de n elementos, `$minN` devuelve todos los elementos del grupo.
- `$minN`compara los datos de entrada siguiendo el orden de comparación de BSON para determinar el tipo de salida apropiado. Cuando los datos de entrada contienen varios tipos de datos, el tipo `$minN` de salida es el más bajo del orden de comparación.

Orden de salida

`$minN`devuelve valores sin ningún orden de clasificación en particular. Si es obligatorio garantizar un orden de clasificación determinado, utilícelo `$bottomN` en su lugar o envuelva el resultado con`$sortArray`.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$minN` acumulador para recuperar las dos puntuaciones más bajas de cada asignatura.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.scores.aggregate([
  { $group: { _id: "$subject", lowestTwo: { $minN: { input: "$score", n: 2 } } } }
]);
```

Salida

```
[
  { "_id": "math", "lowestTwo": [72, 68] },
  { "_id": "science", "lowestTwo": [78, 65] }
]
```

Ejemplo de uso de expresiones (MongoDB Shell)

El `$minN` operador también se puede usar como expresión dentro de una `$project` etapa para devolver los N elementos más pequeños de un campo de matriz.

Crea documentos de muestra

```
db.readings.insertMany([
  { _id: 1, sensor: "A", values: [45, 12, 78, 3, 56] },
  { _id: 2, sensor: "B", values: [90, 23, 67, 11, 44] }
]);
```

Ejemplo de consulta

```
db.readings.aggregate([
```

```
  { $project: {
      lowestThree: { $minN: { input: "$values", n: 3 } }
    }}
]);
```

Salida

```
[
  { "_id": 1, "lowestThree": [45, 12, 3] },
  { "_id": 2, "lowestThree": [44, 23, 11] }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$minN` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve una matriz con los valores n más pequeños de los documentos de cada ventana. Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

> **Note**
>
> Cuando se usa como operador de ventana en`$setWindowFields`, `$minN` está limitado a 100 MB de datos intermedios. Una operación que supera este límite devuelve un error.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

En el ejemplo siguiente se dividen los documentos`ticker`, se ordena cada partición y se devuelven los dos precios más bajos desde el principio de la partición hasta el documento actual. `hour`

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        lowestTwoSoFar: {
          $minN: { input: "$price", n: 2 },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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] }
]
```

Cada documento se complementa con `lowestTwoSoFar` los dos `price` valores más pequeños de su ventana. Como se indica en la sección Ordenación de los resultados, el orden de los valores dentro de la matriz devuelta no está garantizado.

Ejemplos de código

Para ver un ejemplo de código sobre el uso $`minN` del operador, elija la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran el uso del acumulador (in$`group`), el uso de expresiones (in$`project`) y el uso del operador de ventana (in$`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 etapa de canalización de `$minute` agregación de Amazon DocumentDB extrae el valor de los minutos de un campo de fecha o marca horaria.

Este operador resulta útil cuando necesita realizar cálculos o agrupaciones basados en fecha y hora dentro de su canalización de agregación.

Parámetros

- `expression`: el campo de fecha o marca horaria del que se extrae el valor de los minutos.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$minute` operador para agrupar los documentos por el valor de minutos extraído del campo de marca de tiempo y contar el número de documentos de cada grupo.

Cree documentos de muestra

```
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") }
]);
```

Ejemplo de consulta

```
db.events.aggregate([
  {
    $group: {
      _id: {
        minute: { $minute: "$timestamp" }
      },
      count: { $count: {} }
    }
  },
  { $sort: { "_id.minute": 1 } }
]);
```

Salida

```
[
  { "_id": { "minute": 30 }, "count": 2 },
  { "_id": { "minute": 31 }, "count": 2 },
  { "_id": { "minute": 32 }, "count": 1 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$minute` comando, elija la pestaña del idioma que desee usar:

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

El $mod operador es un operador aritmético que realiza una operación de módulo en un número. Devuelve el resto de la división de un número por otro. Este operador se utiliza normalmente para determinar si un número es par o impar, o para distribuir elementos en un número finito de grupos.

Parámetros

- `expression1`: La expresión del dividendo.
- `expression2`: La expresión divisoria.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo utilizar el $mod operador para determinar el número de widgets sobrantes cuando se envían paquetes de 100.

Crea documentos de muestra

```
db.widgets.insertMany([
```

```
  { "_id" : 1, "widget" : "A", "count" : 80372 },
  { "_id" : 2, "widget" : "B", "count" : 409282 },
  { "_id" : 3, "widget" : "C", "count" : 60739 }
])
```

Ejemplo de consulta

```
db.widgets.aggregate([
  { $addFields: { leftOver: { $mod: [ "$count", 100 ] } } }
])
```

Salida

```
[
  { "_id" : 1, "widget" : "A", "count" : 80372, "leftOver" : 72 },
  { "_id" : 2, "widget" : "B", "count" : 409282, "leftOver" : 82 },
  { "_id" : 3, "widget" : "C", "count" : 60739, "leftOver" : 39 }
]
```

El resultado muestra el resto `count` dividido por 100 para cada documento, lo que representa el número de widgets sobrantes al enviarse en paquetes de 100.

Ejemplos de código

Para ver un ejemplo de código para usar el $mod comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$month` operador de Amazon DocumentDB devuelve el mes de una fecha como un número entre 1 y 12. Este operador resulta útil para extraer el componente mensual de un campo de fecha y realizar agregaciones y análisis basados en fechas.

Parámetros

- `date_expression`: Esta es la expresión o el campo que contiene la fecha o la marca de tiempo de la que desea extraer el mes.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$month` operador para extraer el mes de un campo de fecha y agrupar los datos por mes.

Cree documentos de muestra

```
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") }
]);
```

Ejemplo de consulta

```
db.sales.aggregate([
  { $group: {
      _id: { month: { $month: "$date" } },
      totalSales: { $sum: "$price" }
    }},
  { $sort: { "_id.month": 1 } }
]);
```

Salida

```
[
  { _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 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$month` comando, elija la pestaña del idioma que desee usar:

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

El `$multiply` operador de Amazon DocumentDB se utiliza para multiplicar los valores de dos o más campos o expresiones. Este operador es particularmente útil para realizar operaciones aritméticas en campos numéricos de documentos. Se puede emplear en varias etapas del proceso de agregación, como `$project` y`$addFields`, para crear nuevos campos o modificar los existentes.

Parámetros

- `expression1`: la primera expresión numérica que se va a multiplicar.
- `expression2`: la segunda expresión numérica que se va a multiplicar.
- `[expression3, ...]`: (opcional) Expresiones numéricas adicionales que se van a multiplicar.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra `$multiply` cómo calcular `bonus_miles` multiplicando `base_miles` y `bonus_rate` para los clientes que utilizaron una tarjeta de crédito para el viaje.

Ejemplos de documentos

```
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 }
]);
```

Ejemplo de consulta

```
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"]
      }
    }
  }
]);
```

Salida

```
[
  { _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 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$multiply` comando, elija la pestaña del idioma que desee usar:

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()
```

$natural

El `$natural` operador de Amazon DocumentDB se utiliza para ordenar los documentos en su orden natural, que es el orden en el que se insertaron en la colección. Esto contrasta con el comportamiento de ordenación predeterminado, que consiste en ordenar los documentos en función de los valores de los campos especificados.

Parámetros

Ninguno

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$natural` operador para ordenar los documentos de una colección en su orden natural.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.people.find({}, { "_id": 1, "name": 1 }).sort({ "$natural": 1 });
```

Salida

```
[
  { "_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 consulta ordena los documentos de la colección en su orden natural, que es el orden en el que se insertaron.

Ejemplos de código

Para ver un ejemplo de código para usar el `$natural` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El operador de $ne agregación compara dos valores y devuelve `true` si no son iguales; de lo contrario, devuelve`false`.

Parámetros

- `expression1`: El primer valor que se va a comparar.
- `expression2`: El segundo valor que se va a comparar.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra el uso del $ne operador para identificar los pedidos con cambios de estado.

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.orders.aggregate([
```

```
  {
    $project: {
      orderId: 1,
      status: 1,
      expectedStatus: 1,
      needsAttention: { $ne: ["$status", "$expectedStatus"] }
    }
  }
]);
```

Salida

```
[
  { _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 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el operador de $ne agregación, elija la pestaña correspondiente al idioma que desee usar:

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

El operador de `$not` agregación realiza una operación NOT lógica en una expresión. Devuelve `true` si la expresión se evalúa como y `false` si la expresión se evalúa como. `false` `true`

Parámetros

- `expression`: La expresión que se va a negar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$not` operador para invertir valores booleanos.

Cree documentos de muestra

```
db.users.insertMany([
  { _id: 1, name: "Alice", active: true },
  { _id: 2, name: "Bob", active: false },
  { _id: 3, name: "Charlie", active: true }
]);
```

Ejemplo de consulta

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      active: 1,
      inactive: { $not: ["$active"] }
    }
  }
]);
```

Salida

```
[
  { _id: 1, name: 'Alice', active: true, inactive: false },
  { _id: 2, name: 'Bob', active: false, inactive: true },
  { _id: 3, name: 'Charlie', active: true, inactive: false }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el operador de `$not` agregación, elija la pestaña correspondiente al idioma que desee usar:

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()
```

$objeto ToArray

El operador de `$objectToArray` agregación de Amazon DocumentDB convierte un objeto (o documento) en una matriz. La entrada del operador es un documento y la salida consiste en un elemento de matriz para cada par campo-valor del documento de entrada. Este operador resulta útil cuando se necesita trabajar con los campos individuales de un documento como una matriz, por ejemplo, cuando se busca el documento con el valor máximo o mínimo de un campo concreto.

Parámetros

- `expression`: la expresión del documento que se va a convertir en una matriz.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$objectToArray` operador para buscar el documento con el inventario máximo para una cadena de tiendas de alquiler de vídeos.

Cree documentos de muestra

```
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
    }
  }
]);
```

Ejemplo de consulta

```
db.videos.aggregate([
  {
    $project: {
      name: 1,
      videos: {
        $objectToArray: "$inventory"
      }
    }
  },
  {
    $unwind: "$videos"
  },
  {
    $group: {
      _id: "$name",
      maxInventory: {
        $max: "$videos.v"
```

```
      }
    }
  }
]);
```

Salida

```
[
  {
    "_id": "Bravemind",
    "maxInventory": 1000
  },
  {
    "_id": "Live Soft",
    "maxInventory": 1000
  },
  {
    "_id": "Romancing the Rock",
    "maxInventory": 500
  },
  {
    "_id": "Top Pilot",
    "maxInventory": 1000
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$objectToArray` comando, elija la pestaña del idioma que desee usar:

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

El operador de `$or` agregación evalúa varias expresiones y devuelve `true` si al menos una expresión se evalúa como. `true` `false`Solo se devuelve si todas las expresiones lo son. `false`

Parámetros

- `expressions`: Un conjunto de expresiones para evaluar.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra el uso del `$or` operador para comprobar si los productos cumplen alguno de los diversos criterios.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.items.aggregate([
  {
    $project: {
      name: 1,
      price: 1,
      onSale: 1,
      goodDeal: {
        $or: [
          { $lt: ["$price", 50] },
          { $eq: ["$onSale", true] }
        ]
      }
    }
  }
]);
```

Salida

```
[
  { _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 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el operador de `$or` agregación, elija la pestaña correspondiente al idioma que desee usar:

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

El `$out` operador de Amazon DocumentDB se utiliza para escribir el resultado de una canalización de agregación en una colección específica.

`$out`debería ser la última etapa de la canalización.

Parámetros

- `output_collection`: el nombre de la colección de salida en la que se escribirán los resultados de la agregación.

Nota: Si la colección ya existe, se sustituirá por los resultados de la etapa de agregación.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$out` operador de Amazon DocumentDB para escribir los resultados de una canalización de agregación en una nueva colección.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.products.aggregate([
  { $group: { _id: "$category", totalPrice: { $sum: "$price" } } },
  { $out: "product_categories" }
])
```

Salida

Ninguno (los resultados se escriben en la colección de resultados).

El proceso de agregación agrupa los productos por categoría y calcula el precio total de los artículos de cada categoría. El `$out` operador escribe los resultados en una nueva colección denominada «product_categories».

Para ver los resultados en la colección de salida:

```
db.product_categories.find()
[
{ "_id" : "Books", "totalPrice" : 125 },
{ "_id" : "Electronics", "totalPrice" : 300 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$out` comando, elija la pestaña correspondiente al idioma que desee usar:

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()
```

percentil $

Nuevo a partir de la versión 8.0.1.

El `$percentile` operador de Amazon DocumentDB calcula los valores percentiles especificados para los datos numéricos. Como acumulador, calcula los valores percentiles de los documentos de un grupo en fase de proceso de agregación. `$group` Como expresión, calcula los percentiles de una matriz de números.

Parámetros

- `input`: expresión que se resuelve en un valor numérico o en una matriz de valores numéricos.
- p: una matriz de valores de percentiles entre 0 y 1, donde cada valor representa un percentil que se debe calcular. Por ejemplo, `[0.25, 0.5, 0.75]` calcula los percentiles 25, 50 y 75.
- `method`: cadena que especifica el método de cálculo. En la actualidad, solo se admite `"approximate"`.

Comportamiento

El `"approximate"` método utiliza el algoritmo t-digest para calcular métricas aproximadas basadas en percentiles. Con conjuntos de datos pequeños, distintos valores de percentiles pueden dar como resultado el mismo resultado. Por ejemplo, con solo 5 valores por grupo, p90 y p99 pueden devolver el valor máximo del grupo. La precisión mejora a medida que aumenta el número de puntos de datos.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$percentile` operador para calcular los percentiles 25 y 75 de las puntuaciones de cada clase.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.students.aggregate([
  { $group: {
      _id: "$class",
      percentiles: { $percentile: { input: "$score", p: [0.25, 0.75], method:
 "approximate" } }
    }}
]);
```

Salida

```
[
  { "_id": "A", "percentiles": [72, 90] },
  { "_id": "B", "percentiles": [75, 88] }
]
```

Ejemplo de uso de expresiones (MongoDB Shell)

El `$percentile` operador también se puede usar como expresión dentro de una `$project` etapa para calcular los percentiles de un campo de matriz.

Cree documentos de muestra

```
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] }
]);
```

Ejemplo de consulta

```
db.surveys.aggregate([
  { $project: {
      quartiles: { $percentile: { input: "$responses", p: [0.25, 0.5, 0.75], method:
 "approximate" } }
    }}
```

```
]);
```

Salida

```
[
  { "_id": 1, "quartiles": [6, 10, 16] },
  { "_id": 2, "quartiles": [5, 9, 15] }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$percentile` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, calcula los valores percentiles especificados para los documentos de cada ventana. Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un documento. `window`

> **Note**
>
> Cuando se usa como operador de ventana en`$setWindowFields`, `$percentile` está limitado a 100 MB de datos intermedios. Una operación que supera este límite devuelve un error.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

El siguiente ejemplo divide los documentos `sensor` y calcula los percentiles 25 y 75 de `reading` todos los documentos de cada partición.

```
db.sensorData.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$sensor",
      output: {
        readingPercentiles: {
          $percentile: { input: "$reading", p: [0.25, 0.75], method: "approximate" },
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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] }
]
```

Cada documento se complementa con `readingPercentiles` los percentiles 25 y 75 aproximados de todos los documentos de `reading` su partición.

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$percentile` operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran el uso del acumulador (in`$group`), el uso de expresiones (in`$project`) y el uso del operador de ventana (in`$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

Introducido en 8.0

El `$pow` operador de Amazon DocumentDB le permite elevar un número a una potencia. Esto puede resultar útil para realizar cálculos exponenciales dentro de su canalización de agregación.

Parámetros

- `<number>`(obligatorio): el número que se va a elevar a una potencia.
- `<exponent>`(obligatorio): la potencia a la que debe elevarse el número.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$pow` operador para calcular el cuadrado de un número.

Cree documentos de muestra

```
db.numbers.insertMany([
  { "_id": 1, "value": 2 },
  { "_id": 2, "value": 3 },
  { "_id": 3, "value": 4 }
]);
```

Ejemplo de consulta

```
db.numbers.aggregate([
  { $addFields: { "square": { $pow: ["$value", 2] } } }
])
```

Salida

```
[
  { "_id": 1, "value": 2, "square": 4 },
  { "_id": 2, "value": 3, "square": 9 },
  { "_id": 3, "value": 4, "square": 16 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el $pow comando, elija la pestaña del idioma que desee usar:

Node.js

Este es un ejemplo del uso del operador $pow en una Node.js aplicación:

```
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

Este es un ejemplo del uso del operador $pow en una aplicación de Python:

```
from pymongo 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

El operador de `$push` agregación devuelve una matriz de todos los valores de una expresión especificada para cada grupo. Por lo general, se usa dentro de la `$group` etapa para acumular valores en una matriz.

Parámetros

- `expression`: la expresión que se va a evaluar para cada documento del grupo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$push` operador para recopilar todos los nombres de productos de cada categoría.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.sales.aggregate([
  {
    $group: {
      _id: "$category",
      products: { $push: "$product" }
    }
  }
```

```
]);
```

Salida

```
[
  { _id: 'Furniture', products: [ 'Desk', 'Chair' ] },
  { _id: 'Electronics', products: [ 'Laptop', 'Mouse', 'Keyboard' ] }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$push` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve una matriz con todos los valores de la expresión especificada para los documentos de cada ventana, ordenados por orden de clasificación. Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

> **Note**
>
> Cuando se usa como operador de ventana en`$setWindowFields`, `$push` está limitado a 100 MB de datos intermedios. Una operación que supera este límite devuelve un error.

Crea documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

El ejemplo siguiente divide los documentos por`category`, ordena cada partición y devuelve una lista continua de `product` valores desde el principio de la partición hasta el documento actual. `day`

```
db.purchases.aggregate([
  {
```

```
    $setWindowFields: {
      partitionBy: "$category",
      sortBy: { day: 1 },
      output: {
        productsSoFar: {
          $push: "$product",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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" ] }
]
```

Cada documento se complementa con `productsSoFar` la matriz de todos los `product` valores de su partición desde el principio hasta el documento actual, en orden de clasificación.

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$push` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran tanto el uso del acumulador (en`$group`) como el uso del operador de ventanas (en`$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

El `$project` operador de Amazon DocumentDB le permite incluir o excluir campos de los documentos de salida de forma selectiva, pasar valores a la siguiente fase del proceso de procesamiento y calcular nuevos campos a partir de los valores de los documentos de entrada.

Parámetros

- `field`: El campo que se debe incluir o excluir de los documentos de salida, puede ser una ruta de campo (p. ej., «a.b.c»).
- `1otrue`: incluye el campo en la salida.
- `0ofalse`: excluye el campo de la salida.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$project` operador en la colección de estudiantes

Cree documentos de muestra

```
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" }
  ]);
```

Esta consulta incluye solo los `math` campos `name` y de la salida. El `_id` campo se incluye de forma predeterminada a menos que se excluya explícitamente.

```
db.students.aggregate([
  { $project: { "name": 1, "math": 1 } }
])
```

Salida

```
{ _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 }
```

Esta consulta excluye los `_id` campos `grade` y de la salida y muestra todos los demás campos (`name`,`math`,`science`).

```
db.students.aggregate([
  { $project: { "grade": 0, "_id": 0 } }
])
```

Salida

```
{ 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 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$project` comando, elija la pestaña correspondiente al idioma que desee usar:

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()
```

$ radianes ToDegrees

Nuevo a partir de la versión 8.0.1.

El `$radiansToDegrees` operador de Amazon DocumentDB convierte un valor de radianes a grados.

Parámetros

- `expression`: expresión que se resuelve en un valor numérico en radianes.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$radiansToDegrees` operador para convertir los valores en radianes en grados.

Cree documentos de muestra

```
db.angles.insertMany([
  { "_id": 1, "radians": 0 },
  { "_id": 2, "radians": 1.5707963267948966 },
  { "_id": 3, "radians": 3.141592653589793 },
  { "_id": 4, "radians": 6.283185307179586 }
]);
```

Ejemplo de consulta

```
db.angles.aggregate([
  { $project: {
    "degrees": { $radiansToDegrees: "$radians" }
  }}
]);
```

Salida

```
[
```

```
  { "_id": 1, "degrees": 0 },
  { "_id": 2, "degrees": 90 },
  { "_id": 3, "degrees": 180 },
  { "_id": 4, "degrees": 360 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$radiansToDegrees` operador, elige la pestaña del idioma que quieres usar:

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

Novedades de la versión 8.0

El `$rand` operador de Amazon DocumentDB se utiliza para generar un número aleatorio entre 0 y 1.

Parámetros

Ninguno

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$rand` operador para seleccionar aleatoriamente dos documentos de la `temp` colección.

Cree documentos de muestra

```
db.items.insertMany([
  { "name": "pencil", "quantity": 110 },
  { "name": "pen", "quantity": 159 }
])
```

Ejemplo de consulta

```
db.items.aggregate([
  {
    $project: {
```

```
      randomValue: { $rand: {} }
    }
  }
])
```

Salida

```
[
  {
    _id: ObjectId('6924a5edd66dcae121d29517'),
    randomValue: 0.8615243955294392
  },
  {
    _id: ObjectId('6924a5edd66dcae121d29518'),
    randomValue: 0.22815483022099903
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$rand` comando, elija la pestaña del idioma que desee usar:

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

El operador de `$range` agregación de Amazon DocumentDB se utiliza para crear una matriz de números consecutivos dentro de un rango específico. Este operador es particularmente útil para generar secuencias de números, como marcadores de millas para los avituallamientos en una carrera, como se demuestra en los siguientes ejemplos.

Parámetros

- `start`: el valor inicial del rango.

- `end`: el valor final del rango.
- `step`: (opcional) El valor del paso que se utilizará al generar el rango. Si no se proporciona, el valor de paso predeterminado es 1.

Ejemplo (MongoDB Shell)

En este ejemplo, utilizaremos el `$range` operador para generar los marcadores de millas de las estaciones de abastecimiento de agua en una carrera de bicicletas.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.races.aggregate([
  {
    $project: {
      race: 1,
      "waterStations": { $range: [20, "$distance", 20] }
    }
  }
]);
```

Salida

```
[
  {
    _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 ] }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$range` comando, elija la pestaña del idioma que desee usar:

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()
```

$rank

Nuevo a partir de la versión 8.0.2.

El operador de `$rank` ventanas de Amazon DocumentDB devuelve la posición de cada documento dentro de su partición, según el orden de las `$setWindowFields` etapas`sortBy`. Los documentos con el mismo `sortBy` valor reciben la misma clasificación, y el siguiente valor distinto omite las posiciones intermedias, dejando espacios en blanco (por ejemplo,). `1, 2, 2, 4` Se especifica en el `output` campo del `$setWindowFields` escenario.

Es un operador de ventana que se usa solo en el `$setWindowFields` escenario.

Parámetros

- `$rank`toma `{}` como valor un documento vacío.
- No acepta `window` marcos.
- Se necesita exactamente una `sortBy` tecla en el `$setWindowFields` escenario.

Ejemplo (MongoDB Shell)

En el ejemplo siguiente se dividen los documentos`game`, se ordenan las particiones `score` en orden descendente y se asigna un rango a cada jugador.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.players.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$game",
```

```
      sortBy: { score: -1 },
      output: {
        rank: { $rank: {} }
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Dentro de cada `game` partición, los jugadores se clasifican de forma descendente. `score` Bob y Carol empatan `85` y comparten rango`2`, por lo que el siguiente rango pasa a ser `4` para Dave.

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$rank` operador en el `$setWindowFields` escenario, elija la pestaña correspondiente al idioma que desee utilizar:

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 etapa de `$redact` agregación de Amazon DocumentDB se utiliza para incluir o excluir contenido de los documentos de salida de forma selectiva en función de los valores de los campos especificados. Esto resulta especialmente útil en situaciones en las que necesita controlar la visibilidad de los datos confidenciales en función de los niveles de acceso o los permisos de los usuarios.

Parámetros

- `$cond`: una expresión que da como resultado uno de los campos del documento o `$$DESCEND` para cada uno de ellos. `$$KEEP` `$$PRUNE`
- `$$KEEP`: conserva el campo actual del documento de salida.
- `$$PRUNE`: elimina el campo actual del documento de salida.

- `$$DESCEND`: aplica el `$redact` escenario de forma recursiva al campo actual, que es un objeto o una matriz.

Ejemplo (MongoDB Shell)

En este ejemplo, utilizaremos la `$redact` etapa para filtrar los pedidos en función de su estado y mostrar solo los pedidos con valores de estado específicos.

Crea documentos de muestra

```
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" }
])
```

Ejemplo de consulta

```
db.orders.aggregate([
  {
    $redact: {
      $cond: {
        if: { $in: ["$status", ["shipped", "processing"]] },
        then: "$$KEEP",
        else: "$$PRUNE"
      }
    }
  }
])
```

Salida

```
[
  { _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' }
]
```

En este ejemplo, la `$redact` etapa comprueba el valor del `status` campo en cada documento. Si `status` está «enviado» o «procesado», el documento se conserva ($$KEEP). De lo contrario, el documento se borra ($$PRUNE).

Ejemplos de código

Para ver un ejemplo de código para usar el `$redact` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El operador de `$reduce` agregación de Amazon DocumentDB se utiliza para aplicar una función de dos argumentos de forma acumulativa a los elementos de una matriz con el fin de reducir la matriz a un único valor. Este operador es particularmente útil para realizar cálculos o transformaciones complejos en los datos de una matriz dentro del proceso de agregación.

Parámetros

- `input`: La matriz que se va a reducir.
- `initialValue`: El valor inicial que se utilizará en la operación de reducción.
- `in`: la expresión que se va a evaluar en cada elemento de la `input` matriz. Esta expresión debe devolver un valor que se utilizará en la siguiente iteración de la reducción.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$reduce` operador para calcular la suma de todos los elementos de una matriz.

Cree documentos de muestra

```
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] }
])
```

Ejemplo de consulta

```
db.orders.aggregate([
  {
    $project: {
      total: {
        $reduce: {
          input: "$items",
          initialValue: 0,
          in: { $add: ["$$value", "$$this"] }
        }
      }
    }
  }
])
```

Salida

```
[
  { "_id": 1, "total": 15 },
  { "_id": 2, "total": 60 },
  { "_id": 3, "total": 80 },
  { "_id": 4, "total": 300 }
]
```

El `$reduce` operador itera sobre la `items` matriz, añadiendo cada elemento a `initialValue` 0. El resultado es la suma de todos los elementos de la matriz.

Ejemplos de código

Para ver un ejemplo de código para usar el `$reduce` comando, elija la pestaña correspondiente al idioma que desee usar:

Node.js

Este es un ejemplo del uso del `$reduce` operador en una Node.js aplicación:

```
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

Este es un ejemplo del uso del `$reduce` operador en una aplicación de 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

Nuevo a partir de la versión 5.0.

No es compatible con el clúster de Elastic.

El `$regexFind` operador de Amazon DocumentDB se utiliza para realizar coincidencias de expresiones regulares en los campos de cadenas de texto de los documentos. Le permite buscar y extraer subcadenas específicas que coincidan con un patrón de expresión regular determinado.

Parámetros

- `input`: El campo de cadena o la expresión que se va a buscar.
- `regex`: el patrón de expresión regular que debe coincidir.

- `options`: (opcional) Un objeto que especifica parámetros opcionales para la expresión regular, como la distinción entre mayúsculas y minúsculas y la coincidencia de líneas múltiples. Las opciones admitidas son `i` (no distingue entre mayúsculas y minúsculas) y `m` (multilínea).

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$regexFind` operador para buscar documentos en los que el `name` campo coincide con un patrón de expresión regular específico.

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.users.aggregate([
  {
    $project: {
      names: {
        $regexFind: { input: '$name', regex: 'j', options: 'i' }
      }
    }
  },
  { $match: {names: {$ne: null}}}
])
```

Esta consulta devolverá todos los documentos en los que el `name` campo contenga la letra «j» (no distingue entre mayúsculas y minúsculas).

Salida

```
[
  { _id: 1, names: { match: 'J', idx: 0, captures: [] } }
]
```

Nota: Si su consulta utiliza la versión 1 del planificador de Amazon DocumentDB, debe utilizar una sugerencia para utilizar un índice. Sin una pista, la consulta puede realizar un escaneo de la colección. Para comprobar la versión de su planificador y obtener más información sobre el uso de las sugerencias, consulte Planificador de consultas v2.

Ejemplos de código

Para ver un ejemplo de código para usar el `$regexFind` comando, elija la pestaña correspondiente al idioma que desee usar:

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()
```

$ expresión regular FindAll

Introducido en 5.0

El `$regexFindAll` operador de Amazon DocumentDB se utiliza para realizar coincidencias de expresiones regulares en los campos de cadenas de texto de los documentos. Le permite buscar y extraer subcadenas específicas que coincidan con un patrón de expresión regular determinado y devuelve todas las coincidencias de la expresión regular.

Parámetros

- `input`: el campo de cadena o la expresión que se va a buscar.
- `regex`: el patrón de expresión regular que debe coincidir.
- `options`: (opcional) Un objeto que especifica parámetros opcionales para la expresión regular, como la distinción entre mayúsculas y minúsculas y la coincidencia de líneas múltiples. Las opciones admitidas son `i` (no distingue entre mayúsculas y minúsculas) y `m` (multilínea).

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$regexFindAll` operador para extraer todas las secuencias de letras del `email` campo.

Cree documentos de muestra

```
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" }

]);
```

Ejemplo de consulta

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      emailMatches: {
        $regexFindAll: { input: '$email', regex: '[a-z]+', options: 'i' }
      }
    }
  }
])
```

Salida

```
[
  {
    _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: Si su consulta utiliza la versión 1 del planificador de Amazon DocumentDB, debe utilizar una sugerencia para utilizar un índice. Sin una pista, la consulta puede realizar un escaneo de la colección. Para comprobar la versión de su planificador y obtener más información sobre el uso de las sugerencias, consultePlanificador de consultas v2.

Ejemplos de código

Para ver un ejemplo de código para usar el `$regexFindAll` comando, elija la pestaña correspondiente al idioma que desee usar:

Node.js

A continuación, se muestra un ejemplo del uso `$regexFind` del operador en una Node.js aplicación:

```
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

Este es un ejemplo del uso del `$regexFind` operador en una aplicación de 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

Nuevo desde la versión 5.0. No es compatible con el clúster de Elastic.

El `$regexMatch` operador de Amazon DocumentDB se utiliza para realizar coincidencias de expresiones regulares en campos de cadenas. Devuelve un valor booleano (`trueofalse`) que indica si la cadena de entrada coincide con el patrón especificado.

Parámetros

- `input`: La cadena que se comparará con la expresión regular.
- `regex`: El patrón de expresión regular que debe coincidir.
- `options`: (Opcional) Banderas para modificar el comportamiento de la expresión regular, como la coincidencia sin distinguir entre mayúsculas y minúsculas (`i`) o la coincidencia multilínea (). `m`

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$regexMatch` operador para comprobar si los nombres comienzan por la letra «M». El operador devuelve `true` o `false` para cada documento.

Cree documentos de muestra

```
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" },

]);
```

Ejemplo de consulta

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      startsWithM: {
```

```
        $regexMatch: {
          input: "$name",
          regex: "^M",
          options: "i"
        }
      }
    }
  }
]);
```

Salida

```
{ _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 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$regexMatch` comando, elija la pestaña del idioma que desee usar:

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()
```

$ReplaceAll

Introducido en 5.0

El `$replaceAll` operador de Amazon DocumentDB se utiliza para reemplazar todas las apariciones de un patrón de cadenas especificado dentro de un campo por una cadena nueva. Este operador puede resultar útil para tareas como la normalización de datos, la limpieza de texto y la manipulación de cadenas.

Parámetros

- `input`: el campo o la expresión que contiene la cadena que se va a reemplazar.
- `find`: el patrón de cadenas que se va a buscar y reemplazar.
- `replacement`: La cadena con la que se van a reemplazar las apariciones coincidentes.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$replaceAll` operador en una canalización de agregación para reemplazar todas las apariciones de la cadena «Chocolatier» por «Chocolate Co». en el campo «Nombre de la marca» de una colección de «productos».

Cree documentos de muestra

```
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
  }
]);
```

Ejemplo de consulta

```
db.products.aggregate([
  {
    $addFields: {
      "brandName": {
        $replaceAll: {
          input: "$brandName",
          find: "Chocolatier",
          replacement: "Chocolate Co."
        }
      }
    }
  }
])
```

Salida

```
[
  {
    _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
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$replaceAll` comando, elija la pestaña del idioma que desee usar:

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()
```

$ ReplaceOne

Introducido en 5.0

El `$replaceOne` operador de Amazon DocumentDB es un operador de expresión de cadenas que se utiliza en las canalizaciones de agregación para reemplazar la primera aparición de una

subcadena especificada dentro de una cadena por una cadena de reemplazo. Este operador distingue entre mayúsculas y minúsculas y solo reemplaza la primera coincidencia encontrada.

Parámetros

- `input`: La cadena (campo) en la que se va a realizar la búsqueda.
- `find`: La cadena que se va a buscar en la entrada.
- `replacement`: La cadena que sustituirá a la primera aparición de la búsqueda en la entrada (campo).

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$replaceOne` operador dentro de una canalización de agregación para reemplazar las subcadenas en los nombres de los productos.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de agregación

```
db.products.aggregate([
  {
    $addFields: {
      standardizedName: {
        $replaceOne: {
          input: "$name",
          find: "Pack",
          replacement: "Package"
        }
      }
    }
  }
]);
```

Salida

El resultado muestra que solo la primera vez que aparece «Pack» en cada nombre de producto se ha sustituido por «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"
  }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$replaceOne` comando, selecciona la pestaña correspondiente al idioma que deseas usar:

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

El `$replaceRoot` operador se utiliza para reemplazar el documento raíz por el documento incrustado especificado. Esto resulta útil cuando se quiere promover un documento anidado al nivel superior o reestructurar la salida de datos.

Parámetros

- `newRoot`: el nuevo documento raíz que sustituirá al documento raíz existente.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo extraer la información de las direcciones de envío de los pedidos de los clientes, lo que resulta útil para generar etiquetas de envío o listas de direcciones.

Crea documentos de muestra

```
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 } ])
```

Ejemplo de consulta

```
db.orders.aggregate([
  {
    $replaceRoot: {
      newRoot: "$shippingAddress"
    }
  }
])
```

Salida

```
{
    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'
  }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$replaceRoot` comando, elija la pestaña del idioma que desee usar:

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()
```

$ Reemplazar con

Nuevo desde la versión 8.0

No es compatible con el clúster de Elastic.

La etapa de `$replaceWith` agregación de Amazon DocumentDB se utiliza para reemplazar el documento de entrada por un documento nuevo. Todos los campos existentes en el documento de entrada, incluido el campo _id, se sustituyen por el nuevo documento. `$replaceWith`se suele utilizar para aplanar documentos o para llevar un documento incrustado al nivel superior.

Parámetros

- `<replacement>`(obligatorio): el nuevo documento que sustituirá al documento existente.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra cómo utilizar el `$replaceWith` operador para reemplazar un documento existente en una colección de Amazon DocumentDB.

Cree documentos de muestra

```
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 ]
  }
```

```
]);
```

Ejemplo de consulta

```
db.restaurants.aggregate([
  { $replaceWith: {
      name: "$name",
      cuisine: "$cuisine",
      rating: { $avg: "$ratings" }
    }
  }
]);
```

Salida

```
[
  {
    name: 'Biryani Adda',
    cuisine: 'Indian',
    rating: 3.4
  },
  {
    name: 'The Burger Spot',
    cuisine: 'American',
    rating: 2.7777777777777777
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$replaceWith` comando, elija la pestaña del idioma que desee usar:

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

El `$reverseArray` operador de Amazon DocumentDB se utiliza para invertir los elementos de una matriz en el orden especificado. Este operador resulta útil cuando necesita reordenar los elementos de una matriz en la dirección inversa.

Parámetros

- `expression`: La expresión matricial que se va a invertir.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$reverseArray` operador para invertir el orden de los elementos de una matriz.

Cree documentos de muestra

```
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]}
]);
```

Ejemplo de consulta

```
db.miles.aggregate([
  {
    $project: {
      _id: 1,
      member_since: 1,
      credit_card: 1,
      reversed_flight_miles: { $reverseArray: "$flight_miles" }
    }
  }
]);
```

Salida

```
{ "_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 ] }
```

En este ejemplo, el `$reverseArray` operador se utiliza para invertir el orden de la `flight_miles` matriz. El `reversed_flight_miles` campo resultante de la salida muestra los elementos de la matriz en el orden inverso.

Ejemplos de código

Para ver un ejemplo de código para usar el `$reverseArray` comando, elija la pestaña correspondiente al idioma que desee usar:

Node.js

A continuación, se muestra un ejemplo del uso `$reverseArray` del operador en una Node.js aplicación:

```
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

Este es un ejemplo del uso del `$reverseArray` operador en una aplicación de 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

Nuevo desde la versión 8.0.1.

El `$round` operador de Amazon DocumentDB redondea un número a un decimal específico. Cuando un valor se encuentra exactamente a la mitad entre los dos valores más cercanos en ese lugar, se `$round` redondea al dígito par más cercano. Por ejemplo, 2,5 se redondea a 2 y 3,5 se redondea a 4.

Parámetros

- `number`: expresión que se resuelve en un número.
- `place`: opcional. Expresión entera entre -20 y 100 que especifica el número de decimales al que se debe redondear. Un valor negativo se redondea a la izquierda del decimal. El valor predeterminado es 0.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$round` operador para redondear los valores numéricos a un decimal.

Redondear con un valor posicional positivo

```
db.measurements.insertMany([
  {_id: 1, value: 3.456},
  {_id: 2, value: 7.891},
  {_id: 3, value: 12.345}
]);
```

Ejemplo de consulta

```
db.measurements.aggregate([
  { $project: { rounded: { $round: ["$value", 1] } } }
]);
```

Salida

```
[
  {_id: 1, rounded: 3.5},
  {_id: 2, rounded: 7.9},
  {_id: 3, rounded: 12.3}
]
```

Redondear valores intermedios al dígito par más cercano

Cuando un valor se encuentra exactamente a la mitad entre los dos valores más cercanos en el punto objetivo, se `$round` redondea al dígito par más cercano en lugar de redondearlo siempre al alza. En el siguiente ejemplo, se redondean los valores a la mitad a números enteros.

```
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] } } }
]);
```

Salida

```
[
  {_id: 1, rounded: 10},
  {_id: 2, rounded: 12},
  {_id: 3, rounded: 12},
  {_id: 4, rounded: 14}
]
```

Porque `10.5` está exactamente a medio camino entre `10` y`11`, se redondea al valor `10` par más cercano. Del mismo modo, `11.5` y `12.5` ambos se redondean al valor `12` par más cercano y se `13.5` redondean a`14`.

Redondea con un valor posicional negativo

Si `place` es negativo, se `$round` redondea a la izquierda del decimal. En el siguiente ejemplo, se redondean los valores con el primer dígito situado a la izquierda del decimal.

```
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] } } }
]);
```

Salida

```
[
  {_id: 1, rounded: 20},
  {_id: 2, rounded: 30},
  {_id: 3, rounded: 30}
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$round` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar:

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

Nuevo desde la versión 4.0.

No es compatible con el clúster de Elastic.

El `$rtrim` operador de Amazon DocumentDB se utiliza para eliminar los caracteres finales de una cadena. De forma predeterminada, elimina los espacios en blanco finales, pero también puede especificar un conjunto de caracteres para eliminarlos pasando el argumento chars.

Parámetros

- `input`: la cadena de entrada de la que se van a eliminar los espacios en blanco finales.
- `chars`: (opcional) Para eliminar caracteres específicos.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra cómo utilizar el `$rtrim` operador para eliminar los últimos caracteres especificados («*») de una cadena.

Cree documentos de muestra

```
db.collection.insert([
  { "name": "John Doe*  ", "age": 30 },
  { "name": "Jane Smith  ", "age": 25 },
  { "name": "Bob Johnson", "age": 35 }
]);
```

Ejemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
      _id: 0,
      name: { $rtrim: { input: "$name", chars: " *" } },
      age: 1
    }
  }
]);
```

Salida

```
[
  { age: 30, name: 'John Doe' },
  { age: 25, name: 'Jane Smith' },
  { age: 35, name: 'Bob Johnson' }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$rtrim` comando, elija la pestaña del idioma que desee usar:

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 etapa de `$sample` agregación de Amazon DocumentDB se utiliza para seleccionar aleatoriamente un número específico de documentos de una colección. Esto resulta útil para tareas como el análisis de datos, las pruebas y la generación de muestras para su posterior procesamiento.

Parámetros

- `size`: el número de documentos que se van a seleccionar de forma aleatoria.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$sample` escenario para seleccionar aleatoriamente dos documentos de la `temp` colección.

Cree documentos de muestra

```
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") }
]);
```

Ejemplo de consulta

```
db.temp.aggregate([
   { $sample: { size: 2 } }
])
```

Salida

```
{ "_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") }
```

Como muestran los resultados, 2 de los 10 documentos se muestrearon aleatoriamente. Ahora puede usar estos documentos para determinar un promedio o realizar min/max cálculos.

Ejemplos de código

Para ver un ejemplo de código para usar el `$sample` comando, elija la pestaña correspondiente al idioma que desee usar:

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()
```

$search

Nuevo a partir de la versión 5.0.

El `$search` operador de Amazon DocumentDB se utiliza para proporcionar capacidades de búsqueda de texto.

Parámetros

Ninguno

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$search` operador para realizar una consulta de búsqueda de texto.

Cree documentos de muestra

```
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." }
]);
```

Ejemplo de consulta

```
db.textcollection.find(
  { $text: { $search: "John" } }
);
```

Salida

```
[
  {
    _id: 1,
    name: 'John Doe',
    description: 'This is a sample document about John Doe.'
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$search` comando, elija la pestaña del idioma que desee usar:

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

El `$second` operador de Amazon DocumentDB extrae el componente de segundos de una fecha o marca de tiempo. Se usa para recuperar el valor de segundos de un campo de fecha o marca de tiempo.

Parámetros

- `expression`: el campo de fecha o marca de tiempo del que extraer el valor de los segundos. Esta expresión puede ser una ruta de campo o cualquier expresión válida que dé como resultado una fecha o una marca de tiempo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$second` operador para extraer el componente de segundos de un campo de fecha.

Cree documentos de muestra

```
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") }
]);
```

Ejemplo de consulta

```
db.users.aggregate([{ $project: { name: 1, dobSeconds: { $second: "$dob" } } }])
```

Salida

```
[
  { "_id" : ObjectId("6089a9c306a829d1f8b456a1"), "name" : "John", "dobSeconds" : 45 },
  { "_id" : ObjectId("6089a9c306a829d1f8b456a2"), "name" : "Jane", "dobSeconds" : 59 },
  { "_id" : ObjectId("6089a9c306a829d1f8b456a3"), "name" : "Bob", "dobSeconds" : 0 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$second` comando, elige la pestaña del idioma que quieres usar:

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

Nuevo a partir de la versión 8.0

Elastic Cluster no lo admite.

La fase de `$set` agregación de Amazon DocumentDB le permite agregar nuevos campos o actualizar los valores de los campos existentes en la documentación durante un proceso de agregación.

Parámetros

- `expression`: La expresión que se va a evaluar. Puede ser cualquier expresión de agregación válida, incluidas las referencias de campo y las operaciones aritméticas.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso de la etapa de `$set` agregación para calcular los totales multiplicando el campo por el `quantity` campo. `price`

Cree documentos de muestra

```
db.inventory.insertMany([
  { item: "pencil", quantity: 100, price: 0.24},
  { item: "pen", quantity: 204, price: 1.78 }
]);
```

Ejemplo de agregación

```
db.inventory.aggregate([
  {
    $set: {
      total: { $multiply: ["$quantity", "$price"] }
    }
  }
])
```

Salida

```
[
  {
    _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
  }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$set` comando, elija la pestaña correspondiente al idioma que desea usar:

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

Nuevo desde la versión 4.0.

El `$setDifference` operador de Amazon DocumentDB se utiliza para comparar dos conjuntos y devolver los elementos que se encuentran en el primer conjunto pero no en el segundo. Este operador es útil para encontrar los elementos únicos entre dos conjuntos.

Parámetros

- `firstSet`: El primer conjunto que se va a comparar.
- `secondSet`: El segundo conjunto para comparar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$setDifference` operador para encontrar los elementos únicos entre dos conjuntos.

Cree documentos de muestra

```
db.collection.insertMany([
  { _id: 1, fruits: ["apple", "banana", "cherry", "date"] },
  { _id: 2, fruits: ["banana", "cherry", "date", "elderberry"] }
]);
```

Ejemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
      uniqueFruits: { $setDifference: ["$fruits", ["banana", "cherry", "date"]] }
    }
  }
]);
```

Salida

```
[
  { "_id": 1, "uniqueFruits": ["apple"] },
  { "_id": 2, "uniqueFruits": ["elderberry"] }
]
```

La consulta realiza los siguientes pasos:

1. Utiliza el `$project` escenario para crear un campo nuevo `uniqueFruits` para cada documento.

2. El `$setDifference` operador compara la `fruits` matriz con la matriz `["banana", "cherry", "date"]` y devuelve los elementos únicos de la `fruits` matriz.

Ejemplos de código

Para ver un ejemplo de código para usar el `$setDifference` comando, elija la pestaña correspondiente al idioma que desee usar:

Node.js

Este es un ejemplo de cómo usar el `$setDifference` operador en una Node.js aplicación:

```
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

Este es un ejemplo de cómo usar el `$setDifference` operador en una aplicación de 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

El `$setEquals` operador de Amazon DocumentDB se utiliza para determinar si dos conjuntos son iguales. Compara dos matrices y devuelve `true` si contienen los mismos elementos distintos, independientemente de su orden.

Parámetros

- `expression1`: La primera matriz que se va a comparar.
- `expression2`: La segunda matriz que se va a comparar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$setEquals` operador para comparar dos conjuntos de valores.

Cree documentos de muestra

```
db.collection.insertMany([
  { _id: 1, fruits: ["apple", "banana", "cherry"] },
  { _id: 2, fruits: ["banana", "apple", "cherry"] },
  { _id: 3, fruits: ["apple", "banana", "orange"] }
```

```
])
```

Ejemplo de consulta

```
db.collection.find({
  $expr: {
    $setEquals: ["$fruits", ["apple", "banana", "cherry"]]
  }
})
```

Salida

```
{ "_id" : 1, "fruits" : [ "apple", "banana", "cherry" ] }
{ "_id" : 2, "fruits" : [ "banana", "apple", "cherry" ] }
```

La consulta utiliza el `$setEquals` operador para comparar el `fruits` campo de cada documento con la matriz`["apple", "banana", "cherry"]`. Se devuelven los documentos en los que el `fruits` campo es igual a la matriz de comparación.

Ejemplos de código

Para ver un ejemplo de código para usar el `$setEquals` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$setIntersection` operador de Amazon DocumentDB se utiliza para devolver los elementos comunes entre dos o más matrices. Este operador resulta especialmente útil cuando se trabaja con conjuntos de datos, ya que permite encontrar la intersección de varios conjuntos.

Parámetros

- `array1`: La primera matriz que se interseca.
- `array2`: La segunda matriz que se interseca.
- `arrayN`: (opcional) Matrices adicionales para intersecar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$setIntersection` operador para encontrar los elementos comunes entre dos matrices.

Cree documentos de muestra

```
db.collection.insertMany([
  { _id: 1, colors: ["red", "blue", "green"] },
  { _id: 2, colors: ["blue", "yellow", "orange"] },
  { _id: 3, colors: ["red", "green", "purple"] }
])
```

Ejemplo de consulta

```
db.collection.aggregate([
  { $project: {
      _id: 1,
      commonColors: { $setIntersection: ["$colors", ["red", "blue", "green"]] }
    }
  }
])
```

Salida

```
[
  { "_id": 1, "commonColors": ["red", "blue", "green"] },
```

```
  { "_id": 2, "commonColors": ["blue"] },
  { "_id": 3, "commonColors": ["red", "green"] }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$setIntersection` comando, elija la pestaña del idioma que desee usar:

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()
```

conjunto $ IsSubset

El `$setIsSubset` operador de Amazon DocumentDB se utiliza para determinar si un conjunto de valores es un subconjunto de otro conjunto. Resulta útil para realizar comparaciones y operaciones basadas en conjuntos en campos de matriz.

Parámetros

- `field`: El campo al que se va a aplicar el `$setIsSubset` operador.
- `set`: El conjunto con el que comparar el campo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$setIsSubset` operador para comprobar si el `tags` campo es un subconjunto del conjunto especificado.

Cree documentos de muestra

```
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"] }
```

```
]);
```

Ejemplo de consulta

```
db.products.find({
  $expr: { $setIsSubset: [["tag1", "tag2"], "$tags"] }
})
```

*Nota: * `$setIsSubset` es un operador de agregación y no se puede usar directamente en las consultas find (). En este ejemplo, `$expr` se usa con `find()` para cerrar la brecha entre los operadores de consulta y las expresiones de agregación.

Salida

```
[
  { "_id" : 1, "name" : "Product A", "tags" : [ "tag1", "tag2", "tag3" ] },
  { "_id" : 2, "name" : "Product B", "tags" : [ "tag1", "tag2" ] }
]
```

La consulta devuelve los documentos en los que el `tags` campo es un subconjunto del conjunto`["tag1", "tag2"]`.

Ejemplos de código

Para ver un ejemplo de código para usar el `$setIsSubset` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El operador de `$setUnion` agregación de Amazon DocumentDB se utiliza para combinar dos o más conjuntos de valores y devolver un conjunto que contiene todos los elementos únicos de los conjuntos de entrada. Este operador resulta útil cuando necesita realizar operaciones basadas en conjuntos en campos matriciales de sus documentos.

Parámetros

- `expression1`: expresión que se convierte en una matriz.
- `expression2`: expresión que se convierte en una matriz.
- `expressionN`: Expresiones adicionales que se convierten en matrices (opcional).

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$setUnion` operador para combinar los elementos únicos de dos campos de matriz de una colección.

Cree documentos de muestra

```
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"] }
]);
```

Ejemplo de consulta

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      allInterests: { $setUnion: ["$hobbies", "$skills"] }
    }
  }
]);
```

Salida

```
[
  { "_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" ] }
]
```

En este ejemplo, el `$setUnion` operador se utiliza para combinar los elementos únicos de los campos `hobbies` y `skills` matriz de cada documento de usuario. El `allInterests` campo resultante contiene la unión de todas las aficiones y habilidades únicas de cada usuario.

Ejemplos de código

Para ver un ejemplo de código para usar el `$setUnion` comando, elija la pestaña correspondiente al idioma que desee usar:

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()
```

$conjunto WindowFields

Nuevo a partir de la versión 8.0.2.

El `$setWindowFields` escenario de Amazon DocumentDB realiza operaciones en un intervalo específico de documentos (una ventana) de una partición y devuelve los resultados en función del operador de ventana elegido. La etapa agrupa los documentos de entrada en particiones y, si lo desea, los ordena y, para cada documento, calcula uno o más campos de salida en una ventana con los documentos adyacentes.

Sintaxis

```
{
  $setWindowFields: {
    partitionBy: <expression>,
    sortBy: { <sortField1>: <sortOrder>, ... },
    output: {
      <outputField1>: {
        <windowOperator>: <expression>,
        window: {
          documents: [ <lowerBound>, <upperBound> ],
          range: [ <lowerBound>, <upperBound> ],
          unit: <time unit>
        }
      },
      ...
    }
```

```
  }
}
```

Parámetros

- `partitionBy`: opcional. Una expresión única que se utiliza para agrupar los documentos en particiones. Puede ser una ruta de campo (por ejemplo,`"$state"`) u otra expresión. Si se omite, todos los documentos de entrada pertenecen a una sola partición.
- `sortBy`: opcional. Documento que especifica el campo o los campos por los que se ordenan los documentos de cada partición, en orden ascendente o `{ <field>: -1 }` descendente. `{ <field>: 1 }` `sortBy`La A es obligatoria cuando se utilizan límites de ventana basados en documentos o en rangos, y cuando se utiliza uno de los operadores de clasificación.
- `output`: obligatorio. Un documento que especifica los campos que se van a anexar a cada documento. Puede especificar varios campos de salida, pero cada campo de salida toma exactamente un operador de ventana, su argumento y un `window` documento opcional que define los límites de la ventana:`{ <field>: { <windowOperator>: <expression>, window: { ... } } }`.

Límites de ventana

Dentro de cada `output` campo, si lo desea, puede definir los límites de las ventanas con un `window` documento. Una ventana se puede definir por la posición del documento o por un rango de valores.

- `documents`: especifica la ventana como un número de documentos antes y después del documento actual, en el formulario`[ <lowerBound>, <upperBound> ]`. Cada límite es un desplazamiento entero (por ejemplo, `-1` o2) o una de las palabras clave `"unbounded"` y`"current"`. `"unbounded"`extiende la ventana hasta el principio de la partición cuando se usa como límite inferior, o hasta el final de la partición cuando se usa como límite superior. `"current"`significa el documento actual. Una `documents` ventana requiere un, `sortBy` a menos que ambos límites lo sean`"unbounded"`.
- `range`: especifica la ventana como un rango de valores de `sortBy` campo relativos al documento actual, en el formulario`[ <lowerBound>, <upperBound> ]`. Cada límite es un número que se añade al `sortBy` valor del documento actual o a la `"unbounded"` palabra clave. Una `range` ventana requiere exactamente un `sortBy` campo ascendente y no acepta la `"current"` palabra clave.
- `unit`: opcional. Se usa con `range` para interpretar los límites numéricos de la ventana como un intervalo de tiempo en el que el `sortBy` campo contiene valores de fecha. Amazon

DocumentDB admite las siguientes unidades:`"year"`,`"quarter"`,`"month"`,,`"week"`, `"day"` `"hour"` `"minute"`, `"second"` y. `"millisecond"` Cuando `unit` se especifica, los `range` límites deben ser enteros.

Cuando se omite el `window` documento, la ventana predeterminada es la que abarca `documents: ["unbounded", "unbounded"]` toda la partición.

> **Note**
>
> Cuando se utiliza una `range` ventana en lugar de valores de fecha, se recomienda una duración fija `unit` `"day"` (o inferior) en lugar de,, o `"month"` `"quarter"` `"year"`, cuya longitud depende del calendario puede generar límites de ventana ambiguos.

Operadores de ventanas compatibles

Amazon DocumentDB admite los 26 operadores de ventanas siguientes en el `output` campo del `$setWindowFields` escenario:

- `$sum`
- `$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**
>
> El `$rand` operador no es compatible con ningún operador de ventana en `$setWindowFields` fase.

Ventanas vacías

Cuando una ventana no contiene ningún documento (por ejemplo, una `range` ventana sin valores en el rango), el resultado depende del operador: `$count` `$sum` devuelve`0`;`$push`,,,`$addToSet`, `$firstN` `$lastN` `$minN` `$maxN``$topN`, y devuelve una matriz vacía; `$bottomN` `$percentile` devuelve una matriz de `null` valores (uno por cada percentil solicitado); todos los demás operadores devuelven. `null`

Ejemplo (MongoDB Shell)

En el ejemplo siguiente se dividen los documentos`region`, se ordenan las `day` particiones y se calcula el total acumulado `revenue` desde el principio de la partición hasta el documento actual.

Cree documentos de ejemplo

```
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 }
]);
```

Ejemplo de consulta

```
db.sales.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$region",
      sortBy: { day: 1 },
      output: {
        runningRevenue: {
          $sum: "$revenue",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Cada documento se complementa con `runningRevenue` la suma acumulada de `revenue` dentro de su partición hasta el documento actual, incluido.

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$setWindowFields` escenario, elija la pestaña correspondiente al idioma que desee utilizar:

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

Nuevo a partir de la versión 8.0.1.

El `$sigmoid` operador de Amazon DocumentDB aplica la función sigmoidea logística,`1 / (1 + e^(-x))`, a una entrada numérica. La función asigna cualquier número real a un valor entre 0 y 1, lo que produce una curva. S-shaped Esto resulta útil cuando necesita normalizar los valores en el rango de 0 a 1, por ejemplo, para convertir las puntuaciones brutas en puntuaciones de probabilidad o relevancia.

Parámetros

- `expression`: expresión que se resuelve en un valor numérico.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$sigmoid` operador para convertir los valores logit en probabilidades.

Cree documentos de muestra

```
db.predictions.insertMany([
  {_id: 1, logit: -2},
  {_id: 2, logit: 0},
  {_id: 3, logit: 2}
]);
```

Ejemplo de consulta

```
db.predictions.aggregate([
  { $project: { probability: { $sigmoid: "$logit" } } }
]);
```

Salida

```
[
  {_id: 1, probability: 0.11920292202211755},
  {_id: 2, probability: 0.5},
  {_id: 3, probability: 0.8807970779778823}
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$sigmoid` operador, elige la pestaña del idioma que quieres usar:

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

Nuevo a partir de la versión 8.0.1.

El `$sin` operador de Amazon DocumentDB devuelve el seno de un valor que se mide en radianes. Utilícelo en el proceso de agregación para realizar cálculos trigonométricos en campos numéricos.

Parámetros

- `expression`: expresión que se resuelve en un número en radianes.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$sin` operador para calcular el seno de los valores del ángulo en radianes.

Cree documentos de muestra

```
db.angles.insertMany([
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.5235987755982988 },
  { "_id": 3, "angle": 1.5707963267948966 }
]);
```

Ejemplo de consulta

```
db.angles.aggregate([
  { $project: {
    "sine": { $sin: "$angle" }
  }}
]);
```

Salida

```
[
  { "_id": 1, "sine": 0 },
```

```
  { "_id": 2, "sine": 0.49999999999999994 },
  { "_id": 3, "sine": 1 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$sin` operador, elige la pestaña del idioma que quieres usar:

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

Nuevo a partir de la versión 8.0.1.

El `$sinh` operador de Amazon DocumentDB devuelve el seno hiperbólico de un valor. Úselo en la canalización de agregación para realizar cálculos trigonométricos hiperbólicos en campos numéricos.

Parámetros

- `expression`: expresión que se resuelve en un número.

El tipo de devolución es `double` el predeterminado. Si la entrada es un decimal de 128 bits, la salida también es un decimal de 128 bits.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$sinh` operador para calcular el seno hiperbólico de los valores numéricos.

Cree documentos de muestra

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Ejemplo de consulta

```
db.values.aggregate([
  { $project: {
    "result": { $sinh: "$value" }
  }}
]);
```

Salida

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 1.1752011936438014 },
  { "_id": 3, "result": -1.1752011936438014 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del $sinh operador, elige la pestaña del idioma que quieres usar:

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

En Amazon DocumentDB, el `$skip` operador se utiliza para desplazar el punto de partida de los resultados de la consulta, lo que le permite recuperar un subconjunto específico de los documentos coincidentes. Esto resulta especialmente útil en los escenarios de paginación, en los que desea recuperar las siguientes páginas de resultados.

Parámetros

- `skip`: el número de documentos que se van a omitir antes de devolver los documentos restantes.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$skip` operador para recuperar la segunda página de resultados (documentos 11 a 20) de una colección.

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.collection.find({}, { "name": 1 })
             .skip(10)
             .limit(10);
```

Salida

```
[
  { "_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" }
```

```
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$skip` comando, elija la pestaña del idioma que desee usar:

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

El operador de `$slice` agregación permite devolver un subconjunto de una matriz recorriendo la matriz desde el principio o el final de la matriz. Se utiliza para mostrar un número limitado de elementos de un campo de matriz, como los N elementos superiores o inferiores.

Parámetros

- `array`: El campo de matriz que se va a dividir.
- n: un entero que especifica el número de elementos que se van a devolver. Un valor positivo comienza desde el principio de la matriz, mientras que un valor negativo comienza desde el final de la matriz.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo se utilizan `$slice` para devolver los dos primeros dulces favoritos de cada chef.

Cree documentos de muestra

```
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" ] }
]);
```

Ejemplo de consulta

```
db.sweets.aggregate([
  { $project: { _id: 0, name: 1, topTwoFavorites: { $slice: [ "$favorites", 2 ] } } }
]);
```

Salida

```
[
  { name: 'Alvin', topTwoFavorites: [ 'chocolate', 'cake' ] },
  { name: 'Tom', topTwoFavorites: [ 'donuts', 'pudding' ] },
  { name: 'Jessica', topTwoFavorites: [ 'fudge', 'smores' ] },
  { name: 'Rachel', topTwoFavorites: [ 'ice cream' ] }
]
```

En este ejemplo, el `$slice` operador se utiliza para extraer los dos primeros elementos de la `favorites` matriz de cada documento.

Ejemplos de código

Para ver un ejemplo de código para usar el `$slice` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$size` operador se usa para devolver el recuento de elementos dentro de un campo de matriz. Se puede utilizar para determinar el número de elementos de una matriz almacenada en un documento.

Parámetros

- `field`: La ruta del campo cuyo tamaño de matriz desea devolver.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo usar el `$size` operador para devolver el recuento de equipos que sigue cada usuario.

Cree documentos de muestra

```
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"] }
```

```
]);
```

Ejemplo de consulta

```
db.profiles.aggregate([
  {
    $project: {
      _id: 0,
      name: 1,
      "numberOfTeams": { $size: "$teams" }
    }
  }
])
```

Salida

```
[
  { name: 'John Doe', numberOfTeams: 3 },
  { name: 'Jane Smith', numberOfTeams: 2 },
  { name: 'Bob Johnson', numberOfTeams: 3 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$size` comando, elija la pestaña del idioma que desee usar:

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 etapa de `$sort` agregación ordena los documentos en proceso en función de los valores de campo especificados. Los documentos se organizan en orden ascendente o descendente según los criterios de clasificación proporcionados.

Parámetros

- `field`: el nombre del campo por el que se va a ordenar.
- `order`: 1 Utilícelo en orden ascendente o `-1` descendente.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo se muestra cómo usar la `$sort` etapa para ordenar los productos por precio en orden descendente.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.products.aggregate([
  { $sort: { price: -1 } }
]);
```

Salida

```
[
  { _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 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso de la etapa de `$sort` agregación, elija la pestaña correspondiente al idioma que desea usar:

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()
```

Ordenación incremental

La clasificación incremental es una optimización en la que Amazon DocumentDB utiliza un índice que ya ordena los campos principales de una clasificación de varios campos. A continuación, Amazon DocumentDB ordena solo los campos restantes de cada grupo de documentos que comparten los mismos valores iniciales. En lugar de ordenar todo el conjunto de resultados de la memoria en un bloque (una clasificación completa), Amazon DocumentDB lee los documentos en orden de índice. A continuación, realiza muchas clasificaciones pequeñas e independientes, una por grupo. Los resultados son idénticos a los de una clasificación completa. Solo cambian el plan de consulta y la cantidad de memoria y trabajo que utiliza.

Beneficios

La clasificación incremental ofrece las siguientes ventajas:

- Menor uso de memoria y menos errores de clasificación basados en discos. Amazon DocumentDB solicita un grupo pequeño a la vez en lugar de todo el conjunto de resultados de una sola vez.
- Resultados más rápidos con **`limit`**. Amazon DocumentDB puede emitir documentos ordenados antes de leer toda la entrada. Una clasificación seguida de una `limit` (por ejemplo, una consulta con la letra N superior o paginada) puede detenerse tan pronto como haya generado suficientes documentos.

Cuando Amazon DocumentDB elige una clasificación incremental

Por ejemplo, si se ordena por `{ customerId: 1, orderDate: -1 }` y solo se indica por`{ customerId: 1 }`, el índice ya devuelve los documentos ordenados por`customerId`. Amazon DocumentDB solo necesita ordenar `customerId` por `orderDate` cada grupo de documentos que comparten el mismo nombre. El campo inicial ya ordenado (`customerId`) es la clave preordenada. En función de la clasificación que ya proporcione un índice, Amazon DocumentDB elige uno de los tres comportamientos siguientes:

- El índice ordena cada clave de clasificación. No se necesita ningún tipo de clasificación. Amazon DocumentDB lee los documentos directamente en el orden solicitado.
- El índice ordena el prefijo inicial de las claves de clasificación. Amazon DocumentDB utiliza una clasificación incremental, ordenando solo las claves finales de cada grupo preordenado.
- Ningún índice ordena ninguna clave de clasificación inicial. Amazon DocumentDB realiza una clasificación completa de toda la entrada.

Requisitos y limitaciones

La clasificación incremental requiere Amazon DocumentDB 8.0.2 o una versión posterior, con la versión 3.0 del planificador de consultas predeterminado en Amazon DocumentDB 8.0.2. Para obtener más información sobre las versiones del planificador, consulte. Planificador de consultas v3

Amazon DocumentDB no puede usar un índice parcial o un índice de varias claves (matrices) para proporcionar el prefijo preordenado para una clasificación incremental. En esos casos, Amazon DocumentDB vuelve a un orden completo. Esto nunca afecta a la exactitud de los resultados.

Verificar una clasificación incremental con **`explain()`**

`explain()`Utilícelo para confirmar que una consulta utiliza una clasificación incremental. En el plan ganador, aparece una clasificación incremental como una `SORT` etapa que incluye una `presortedKeys` matriz que enumera los principales campos de clasificación que el índice ya ha ordenado. Una clasificación completa aparece como una `SORT` etapa sin una `presortedKeys` matriz. El `presortedKeys` campo es una DocumentDB-specific adición de Amazon al `explain()` resultado y no aparece en los planes explicativos de 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

Nuevo desde la versión 8.0

No es compatible con el clúster de Elastic.

El `$sortArray` operador de Amazon DocumentDB ordena los elementos de una matriz. En el caso de las matrices de valores escalares, la matriz se ordena por el valor del elemento. En el caso de las matrices de documentos, la matriz se ordena por el campo o los campos especificados. El operador devuelve la matriz ordenada en orden ascendente (1) o descendente (`-1`).

> **Note**
>
> `$sortArray`se admite en la `$project` fase del proceso de agregación con la versión 3 del planificador de consultas.

Sintaxis

```
{ $sortArray: { input: <array>, sortBy: <sort specification> } }
```

Parámetros

- `input`: La matriz que se va a ordenar.
- `sortBy`: El orden de clasificación. Para una matriz de valores escalares, especifique 1 si son ascendentes o `-1` descendentes. Para una matriz de documentos, especifique un documento del formulario `{ field: 1 }` o `{ field: -1 }` para cada campo por el que desee ordenar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$sortArray` operador para ordenar los elementos de una matriz en orden ascendente.

Cree documentos de muestra

```
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]}
]);
```

Ejemplo de consulta

```
db.miles.aggregate([
  {
    $project: {
      _id: 1,
      member_since: 1,
      credit_card: 1,
      sorted_flight_miles: { $sortArray: { input: "$flight_miles", sortBy: 1 } }
    }
  }
]);
```

Salida

```
{ "_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 ] }
```

En este ejemplo, el `$sortArray` operador ordena la `flight_miles` matriz en orden ascendente. El `sorted_flight_miles` campo resultante de la salida muestra los elementos de la matriz ordenados.

Ejemplos de código

Para ver un ejemplo de código para usar el `$sortArray` comando, elija la pestaña correspondiente al idioma que desee usar:

Node.js

A continuación, se muestra un ejemplo del uso `$sortArray` del operador en una Node.js aplicación:

```
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

Este es un ejemplo del uso del `$sortArray` operador en una aplicación de 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()
```

$ordenar ByCount

Nuevo a partir de la versión 8.0.1.

La fase de `$sortByCount` agregación de Amazon DocumentDB agrupa los documentos entrantes en función del valor de una expresión específica, luego calcula el recuento de documentos de cada grupo distinto y ordena los resultados por recuento en orden descendente. Equivale a una `$group` etapa con un `$sum` acumulador seguida de una `$sort` etapa en el campo de recuento.

Parámetros

- `expression`: La expresión por la que se va a agrupar. Puede ser una ruta de campo (con el prefijo$) o cualquier expresión de agregación válida.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$sortByCount` escenario para contar y ordenar los documentos por un campo de categoría.

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.products.aggregate([
  { $sortByCount: "$category" }
]);
```

Salida

```
[
  { "_id": "electronics", "count": 3 },
  { "_id": "food", "count": 2 },
  { "_id": "tools", "count": 1 }
]
```

Los resultados se ordenan automáticamente `count` en orden descendente.

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$sortByCount` escenario, elija la pestaña correspondiente al idioma que desee utilizar:

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

El operador de `$split` agregación de Amazon DocumentDB se utiliza para dividir una cadena en una matriz de subcadenas en función de un delimitador específico. Esto puede resultar útil para analizar campos de cadenas complejos y extraer componentes individuales para su posterior procesamiento.

Parámetros

- `string`: La cadena que se va a dividir.

- `delimiter`: El carácter o la cadena utilizados para dividir la cadena de entrada.

Ejemplo (MongoDB Shell)

En este ejemplo, solemos `$split` separar los componentes de un campo «Escritorio» en una matriz, lo que facilita el procesamiento de los datos.

Crea documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.people.aggregate([
  { $project: { parts: { $split: ["$Desk", "-"] } } }
]);
```

Salida

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

El resultado de `$split` crea una matriz que se puede utilizar en la aplicación para mostrar la información a los empleados.

Ejemplos de código

Para ver un ejemplo de código para usar el `$split` comando, elija la pestaña correspondiente al idioma que desee usar:

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

Nuevo a partir de la versión 4.0.

El `$sqrt` operador de Amazon DocumentDB se utiliza para calcular la raíz cuadrada de un número.

Parámetros

- `expression`: El argumento puede ser cualquier expresión válida siempre que se resuelva en un número no negativo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$sqrt` operador para calcular la raíz cuadrada de un número.

Cree documentos de muestra

```
db.numbers.insertMany([
  { "_id": 1, "number": 16 },
  { "_id": 2, "number": 36 },
  { "_id": 3, "number": 64 }
]);
```

Ejemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    "_id": 1,
    "square_root": { $sqrt: "$number" }
  }}
]);
```

Salida

```
[
  { _id: 1, square_root: 4 },
  { _id: 2, square_root: 6 },
  { _id: 3, square_root: 8 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$sqrt` comando, elija la pestaña del idioma que desee usar:

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()
```

$ estándar DevPop

Nuevo a partir de la versión 8.0.1.

El `$stdDevPop` operador de Amazon DocumentDB calcula la desviación estándar poblacional de los valores numéricos. Como acumulador, calcula la desviación estándar de la población entre los documentos de un grupo en fase de proceso de agregación. `$group` Como expresión, calcula la desviación estándar poblacional de una matriz de números. La desviación estándar de la población utiliza N como divisor (no N-1). Non-numeric los valores se ignoran. Si no hay valores numéricos, se devuelven`null`. Si solo hay un valor numérico, se devuelve. `0`

Parámetros

- `expression`: expresión que se resuelve en un valor numérico o en una matriz de valores numéricos.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$stdDevPop` operador para calcular la desviación estándar poblacional de las puntuaciones por asignatura.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.scores.aggregate([
  { $group: {
      _id: "$subject",
      stdDev: { $stdDevPop: "$score" }
    }}
]);
```

Salida

```
[
  { "_id": "math", "stdDev": 10.75174404457249 },
  { "_id": "science", "stdDev": 12.441864811996632 }
]
```

Ejemplo de uso de expresiones (MongoDB Shell)

El `$stdDevPop` operador también se puede usar como expresión dentro de una `$project` etapa para calcular la desviación estándar poblacional de un campo matricial.

Cree documentos de muestra

```
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] }
]);
```

Ejemplo de consulta

```
db.experiments.aggregate([
  { $project: {
      stdDev: { $stdDevPop: "$measurements" }
    }}
]);
```

Salida

```
[
  { "_id": 1, "stdDev": 2.8284271247461903 },
  { "_id": 2, "stdDev": 0 },
  { "_id": 3, "stdDev": 2.8284271247461903 }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$stdDevPop` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve la desviación estándar poblacional de la expresión especificada para los documentos de cada ventana. Se especifica el operador en el `output` campo y, si lo desea, se definen los límites de la ventana con un `window` documento.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

En el ejemplo siguiente se dividen los documentos`sensor`, se ordena cada partición y se devuelve la desviación estándar acumulada `time` `reading` desde el principio de la partición hasta el documento actual.

```
db.sensorData.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$sensor",
      sortBy: { time: 1 },
      output: {
        runningStdDev: {
          $stdDevPop: "$reading",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Cada documento se complementa con `runningStdDev` la desviación estándar de población `reading` dentro de su partición hasta el documento actual, incluido. Cuando la ventana contiene un único valor, `$stdDevPop` devuelve`0`.

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$stdDevPop` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran el uso del acumulador (in`$group`), el uso de expresiones (in`$project`) y el uso del operador de ventana (in`$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()
```

$ estándar DevSamp

Nuevo a partir de la versión 8.0.1.

El `$stdDevSamp` operador de Amazon DocumentDB calcula la desviación estándar muestral de los valores numéricos. Como acumulador, calcula la desviación estándar de la muestra entre los documentos de un grupo en la fase de proceso de agregación. `$group` Como expresión, calcula la desviación estándar muestral de una matriz de números. La desviación estándar de la muestra N-1 se utiliza como divisor (corrección de Bessel). Non-numeric los valores se ignoran. Si hay menos de dos valores numéricos, se devuelven`null`.

Parámetros

- `expression`: expresión que se resuelve en un valor numérico o en una matriz de valores numéricos.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$stdDevSamp` operador para calcular la desviación estándar muestral de las puntuaciones por asignatura.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.scores.aggregate([
  { $group: {
      _id: "$subject",
      stdDev: { $stdDevSamp: "$score" }
    }}
]);
```

Salida

```
[
  { "_id": "math", "stdDev": 6.454972243679028 },
  { "_id": "science", "stdDev": 6.454972243679028 }
]
```

Ejemplo de uso de expresiones (MongoDB Shell)

El `$stdDevSamp` operador también se puede usar como expresión dentro de una `$project` etapa para calcular la desviación estándar muestral de un campo de matriz.

Cree documentos de muestra

```
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] }
]);
```

Ejemplo de consulta

```
db.experiments.aggregate([
  { $project: {
      stdDev: { $stdDevSamp: "$measurements" }
    }}
]);
```

Salida

```
[
  { "_id": 1, "stdDev": 3.1622776601683795 },
  { "_id": 2, "stdDev": 0 },
  { "_id": 3, "stdDev": 3.1622776601683795 }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$stdDevSamp` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve la desviación estándar muestral de la expresión especificada para los documentos de cada ventana. Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

Cree documentos de muestra

```
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 }
```

```
]);
```

Ejemplo de consulta

En el ejemplo siguiente se dividen los documentos`sensor`, se ordena cada partición y se devuelve la desviación estándar del ejemplo en ejecución `value` desde el principio de la partición hasta el documento actual. `time`

```
db.sensorReadings.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$sensor",
      sortBy: { time: 1 },
      output: {
        runningStdDev: {
          $stdDevSamp: "$value",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Cada documento se complementa con `runningStdDev` la desviación estándar de muestra `value` dentro de su partición hasta el documento actual, incluido. Cuando la ventana contiene menos de dos valores (el primer documento de cada partición), `$stdDevSamp` devuelve`null`.

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$stdDevSamp` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran el uso del acumulador (in`$group`), el uso de expresiones (in`$project`) y el uso del operador de ventana (in`$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

El `$strLenBytes` operador de Amazon DocumentDB se utiliza para determinar la longitud de una cadena en bytes. Esto resulta útil cuando necesita comprender el tamaño de almacenamiento de un campo de cadena, especialmente cuando se trata de caracteres Unicode que pueden utilizar más de un byte por carácter.

Parámetros

- `expression`: la expresión de cadena cuya longitud se calcula.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo utilizar el `$strLenBytes` operador para calcular la longitud de los campos de cadena en bytes.

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.people.aggregate([
  {
    $project: {
      "Desk": 1,
      "length": { $strLenBytes: "$Desk" }
    }
  }
])
```

Salida

```
{ "_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 }
```

Tenga en cuenta que la longitud de la cadena BVV-021 «Düsseldorf-» es de 19 bytes, que es diferente del número de puntos de código (18) debido a que el carácter Unicode «Ü» ocupa 2 bytes.

Ejemplos de código

Para ver un ejemplo de código para usar el `$strLenBytes` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$strLenCP` operador de Amazon DocumentDB se utiliza para determinar la longitud de una expresión de cadena en puntos de código (caracteres Unicode). Esto resulta útil cuando necesita saber el número de caracteres de una cadena, en lugar del número de bytes.

Parámetros

- `expression`: la expresión de cadena cuya longitud se debe devolver en puntos de código.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$strLenCP` operador para determinar la longitud de las cadenas con caracteres Unicode.

Cree documentos de muestra

```
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" }
])
```

Ejemplo de consulta

```
db.people.aggregate([
  {
    $project: {
      "Desk": 1,
      "length": { $strLenCP: "$Desk" }
    }
```

```
  }
])
```

Salida

```
{ "_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 }
```

Observe la diferencia en la medida de longitud de la cadena «Düsseldorf-BVV-021», que contiene un carácter Unicode (Ü). El `$strLenCP` operador cuenta correctamente el número de caracteres Unicode, mientras que el `$strLenBytes` operador cuenta el número de bytes.

Ejemplos de código

Para ver un ejemplo de código para usar el `$strLenCP` comando, elija la pestaña correspondiente al idioma que desee usar:

Node.js

A continuación, se muestra un ejemplo del uso del `$strLenCP` operador en una Node.js aplicación con el controlador 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

Este es un ejemplo del uso del `$strLenCP` operador en una aplicación de Python con el PyMongo controlador:

```
from pymongo 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

El `$strcasecmp` operador de Amazon DocumentDB realiza una comparación entre dos cadenas que no distingue mayúsculas de minúsculas. Devuelve un valor entero que indica la comparación lexicográfica de las dos cadenas de entrada, ignorando las diferencias entre mayúsculas y minúsculas.

Parámetros

- `string1`: La primera cadena que se va a comparar.
- `string2`: La segunda cadena que se va a comparar.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo utilizar el `$strcasecmp` operador para comparar las cadenas de ubicación de los escritorios de una `people` colección, ignorando las diferencias entre mayúsculas y minúsculas.

Cree documentos de muestra

```
db.people.insertMany([
  { "_id": 1, "Desk": "mke233-wi" },
  { "_id": 2, "Desk": "MKE233-WI" },
  { "_id": 3, "Desk": "mke233-wi" }
]);
```

Ejemplo de consulta

```
db.people.aggregate([
  {
    $project: {
      item: 1,
      compare: { $strcasecmp: ["$Desk", "mke233-wi"] }
    }
  }
]);
```

Salida

```
{ "_id" : 1, "compare" : 0 }
{ "_id" : 2, "compare" : 0 }
{ "_id" : 3, "compare" : 0 }
```

El resultado muestra que la comparación entre el `"Desk"` campo y la cadena se `"mke233-wi"` devuelve `0` para los tres documentos, lo que indica que las cadenas son iguales si se omiten las mayúsculas y minúsculas.

Ejemplos de código

Para ver un ejemplo de código para usar el `$strcasecmp` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$substr` operador de Amazon DocumentDB se utiliza para extraer una subcadena de una cadena determinada. Resulta especialmente útil cuando necesita definir subcadenas en función de un rango de caracteres, en lugar de un rango de bytes. Esto es especialmente importante cuando se trata de cadenas Unicode, donde el número de bytes utilizados para representar un carácter puede variar.

Parámetros

- `string`: la cadena de entrada de la que se extrae la subcadena.
- `start`: la posición inicial (basada en cero) de la subcadena que se va a extraer. Puede ser una expresión entera no negativa.
- `length`: el número de caracteres de la subcadena extraída. Puede ser una expresión entera no negativa.

Ejemplo (MongoDB Shell)

En este ejemplo, demostraremos el uso de `$substr` para extraer la abreviatura de estado de la ubicación del escritorio de un empleado.

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.people.aggregate([
  {
    $project: {
      "state": { $substr: ["$Desk", 12, 3] }
    }
  }
])
```

Salida

```
{ "_id": 1, "state": "NRW" },
{ "_id": 2, "state": "HBB" },
{ "_id": 3, "state": "SHH" },
{ "_id": 4, "state": "BBB" }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$substr` comando, elija la pestaña del idioma que desee usar:

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

El `$substrBytes` operador de Amazon DocumentDB se utiliza para extraer una subcadena de una cadena en función de un rango de bytes especificado. Este operador resulta útil cuando necesita extraer una subcadena de una cadena y el número de bytes necesario para representar cada carácter de la cadena es importante.

A diferencia `$substrCP` de lo que ocurre con el número de puntos de código Unicode, `$substrBytes` funciona con el número de bytes necesarios para representar los caracteres de la cadena. Esto puede resultar especialmente útil cuando se trabaja con cadenas que contienen caracteres que no son ASCII, ya que es posible que se necesite más de un byte para representarlos.

*Nota: * `$substr` ha quedado obsoleto desde la versión 3.4. `$substr`ahora es un alias de. `$substrBytes`

Parámetros

- `string`: La cadena de entrada de la que se extraerá la subcadena.
- `startByte`: la posición del byte inicial basada en cero de la subcadena que se va a extraer. Se puede usar un valor negativo para especificar una posición desde el final de la cadena.
- `length`: el número de bytes de la subcadena que se va a extraer.

Ejemplo (MongoDB Shell)

En este ejemplo, utilizaremos `$substrBytes` para extraer una subcadena de una cadena que contiene caracteres que no son ASCII.

Crea documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.people.aggregate([
  {
    $project: {
      "state": { $substrBytes: [ "$Desk", 12, 3] }
    }
  }
])
```

Salida

```
{ "_id": 1, "state": "NRW" },
{ "_id": 2, "state": "HBB" },
{ "_id": 3, "state": "SHH" },
{ "_id": 4, "state": "BBB" }
```

En este ejemplo, solemos `$substrBytes` extraer una subcadena de 3 bytes a partir del byte 12 del `Desk` campo. Esto nos permite extraer la abreviatura de estado de 2 caracteres, aunque la cadena pueda contener caracteres que no son ASCII.

Ejemplos de código

Para ver un ejemplo de código para usar el `$substrBytes` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$substrCP` operador de Amazon DocumentDB se utiliza para extraer una subcadena de una cadena, donde la subcadena se especifica como un rango de puntos de UTF-8 código (CP). Este operador resulta especialmente útil cuando se trabaja con cadenas Unicode, ya que permite extraer subcadenas sin tener que preocuparse por la representación subyacente de los caracteres en bytes.

A diferencia del `$substrBytes` operador, que opera en posiciones de bytes, el `$substrCP` operador trabaja con posiciones de puntos de código. Esto facilita el trabajo con cadenas que contienen caracteres que no son ASCII, ya que es posible que el número de puntos de código no coincida con el número de bytes o caracteres.

Parámetros

- `string`: la cadena de entrada de la que se extrae la subcadena.
- `start`: la posición del punto de código inicial (de base cero) desde la que se extrae la subcadena.
- `length`: el número de puntos de código que se van a extraer.

Ejemplo (MongoDB Shell)

En este ejemplo, utilizaremos el `$substrCP` operador para extraer la abreviatura de estado de una cadena que contiene la ubicación del escritorio del empleado.

Crea documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.people.aggregate([
  {
    $project: {
      "state": { $substrCP: ["$Desk", 25, 2] }
    }
  }
]);
```

Salida

```
{ "_id" : 1, "state" : "MN" }
{ "_id" : 2, "state" : "LA" }
{ "_id" : 3, "state" : "CA" }
```

En este ejemplo, sabemos que la abreviatura del estado comienza en el punto de código número 25 del `Desk` campo y tiene una longitud de 2 puntos de código. Al usar el `$substrCP` operador, podemos extraer la abreviatura del estado sin tener que preocuparnos por la representación en bytes subyacente de la cadena.

Ejemplos de código

Para ver un ejemplo de código para usar el `$substrCP` comando, elija la pestaña correspondiente al idioma que desee usar:

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()
```

Tanto en el Node.js ejemplo como en Python, utilizamos el `$substrCP` operador para extraer la abreviatura de estado del `Desk` campo, de forma similar al ejemplo de MongoDB Shell.

$subtract

El `$subtract` operador de Amazon DocumentDB se utiliza para restar valores. Se puede usar para restar fechas, números o una combinación de ambos. Este operador es útil para calcular la diferencia entre dos fechas o restar un valor de un número.

Parámetros

- `expression1`: El primer valor que se va a restar.
- `expression2`: El segundo valor del que se va a restar. `<expression1>`

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$subtract` operador para calcular la diferencia entre dos fechas.

Cree un documento de muestra

```
db.dates.insert([
  {
  "_id": 1,
  "startDate": ISODate("2023-01-01T00:00:00Z"),
  "endDate": ISODate("2023-01-05T12:00:00Z")
  }
]);
```

Ejemplo de consulta

```
db.dates.aggregate([
  {
    $project: {
      _id: 1,
      durationDays: {
        $divide: [
          { $subtract: ["$endDate", "$startDate"] },
          1000 * 60 * 60 * 24  // milliseconds in a day
        ]
```

```
      }
    }
  }
]);
```

Salida

```
[ { _id: 1, durationDays: 4.5 } ]
```

En este ejemplo, el `$subtract` operador se utiliza para calcular la diferencia entre `$endDate` y `$startDate` en días.

Ejemplos de código

Para ver un ejemplo de código para usar el `$subtract` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$sum` operador de Amazon DocumentDB devuelve la suma de la expresión especificada para cada documento de un grupo. Es un operador acumulador de grupos que se suele utilizar en la fase $group de una canalización de agregación para realizar cálculos de suma.

Parámetros

- `expression`: la expresión numérica que se va a sumar. Puede ser una ruta de campo, una expresión o una constante.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$sum` operador para calcular las ventas totales de cada producto.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.sales.aggregate([
```

```
  { $group: {
      _id: "$product",
      totalSales: { $sum: { $multiply: [ "$price", "$quantity" ] } }
    }}
]);
```

Salida

```
[
  { "_id": "abc", "totalSales": 50 },
  { "_id": "xyz", "totalSales": 120 }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$sum` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve la suma de la expresión especificada para los documentos de cada ventana. Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

El ejemplo siguiente divide los documentos por`product`, ordena cada partición por `day` y devuelve un total acumulado de `amount` desde el principio de la partición hasta el documento actual.

```
db.dailySales.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$product",
      sortBy: { day: 1 },
```

```
      output: {
        runningTotal: {
          $sum: "$amount",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Cada documento se complementa con `runningTotal` la suma acumulada de `amount` dentro de su partición hasta el documento actual, incluido.

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$sum` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran tanto el uso del acumulador (en`$group`) como el uso del operador de ventanas (en`$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

Nuevo desde la versión 4.0.

No es compatible con el clúster de Elastic.

El `$switch` operador es un operador de expresión condicional de Amazon DocumentDB que le permite evaluar una lista de expresiones de mayúsculas y minúsculas y devolver el valor del primer caso que se evalúe como verdadero o un valor predeterminado si ninguna expresión de mayúsculas y minúsculas es verdadera.

Parámetros

- `branches`: matriz de documentos, cada uno de los cuales tiene un campo de mayúsculas y minúsculas que contiene la expresión booleana que se va a evaluar y un campo que contiene el valor que se debe devolver si la expresión de mayúsculas y minúsculas es verdadera.
- `default`: (opcional) El valor que se devolverá si ninguna de las expresiones entre mayúsculas y minúsculas es verdadera.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$switch` operador para determinar el coste de envío de un pedido en función del total del pedido.

Cree documentos de muestra

```
db.orders.insertMany([
  { _id: 1, total: 50 },
  { _id: 2, total: 150 },
  { _id: 3, total: 250 }
]);
```

Ejemplo de consulta

```
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
        }
      }
    }
  }
])
```

Salida

```
[
  {
    "_id": 1,
    "total": 50,
    "shippingCost": 5
  },
  {
    "_id": 2,
    "total": 150,
    "shippingCost": 10
  },
  {
    "_id": 3,
    "total": 250,
    "shippingCost": 15
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$switch` comando, elija la pestaña del idioma que desee usar:

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

Nuevo a partir de la versión 8.0.1.

El `$tan` operador de Amazon DocumentDB devuelve la tangente de un valor que se mide en radianes.

La expresión de entrada debe convertirse en un valor numérico. Si el valor está en grados, aplíquelo `$degreesToRadians` antes de pasarlo a`$tan`.

El tipo de devolución es `double` el predeterminado. Si la entrada es un decimal de 128 bits, la salida también es un decimal de 128 bits.

Parámetros

- `expression`: expresión que se resuelve en un número en radianes.

Comportamiento

null, NaN y +/- Infinity

Ejemplo	Resultados
`{ $tan: NaN }`	`NaN`
`{ $tan: null }`	`null`
`{ $tan: Infinity }` o `{ $tan: -Infinity }`	Lanza un error.

Cuando la entrada es `null` o falta el campo al que se hace referencia, `$tan` devuelve`null`. Una entrada de `NaN` productos`NaN`. El infinito positivo o negativo provoca un error porque la tangente no está definida en esos valores.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$tan` operador para calcular la tangente de un ángulo en radianes.

Cree documentos de ejemplo

```
db.angles.insertMany([
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.7853981633974483 },
  { "_id": 3, "angle": -0.7853981633974483 }
]);
```

Ejemplo de consulta

```
db.angles.aggregate([
  { $project: {
    "tangent": { $tan: "$angle" }
  }}
]);
```

Salida

```
[
  { "_id": 1, "tangent": 0 },
  { "_id": 2, "tangent": 0.9999999999999999 },
  { "_id": 3, "tangent": -0.9999999999999999 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$tan` operador, elige la pestaña del idioma que quieres usar:

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

Nuevo a partir de la versión 8.0.1.

El `$tanh` operador de Amazon DocumentDB devuelve la tangente hiperbólica de un valor.

La expresión de entrada debe convertirse en un valor numérico.

El tipo de devolución es `double` el predeterminado. Si la entrada es un decimal de 128 bits, la salida también es un decimal de 128 bits.

Parámetros

- `expression`: expresión que se resuelve en un número.

Comportamiento

null, NaN y +/- Infinity

Ejemplo	Resultados
`{ $tanh: NaN }`	`NaN`
`{ $tanh: null }`	`null`
`{ $tanh: -Infinity }`	`-1`
`{ $tanh: Infinity }`	`1`

Cuando la entrada es `null` o falta el campo al que se hace referencia, `$tanh` devuelve`null`. Una entrada de `NaN` productos`NaN`. Para el infinito positivo, el resultado es`1`; para el infinito negativo, el resultado es`-1`.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$tanh` operador para calcular la tangente hiperbólica de un valor.

Cree documentos de muestra

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Ejemplo de consulta

```
db.values.aggregate([
  { $project: {
    "result": { $tanh: "$value" }
  }}
]);
```

Salida

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 0.7615941559557649 },
  { "_id": 3, "result": -0.7615941559557649 }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$tanh` operador, elige la pestaña del idioma que quieres usar:

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()
```

$superior

Nuevo a partir de la versión 8.0.1.

Utilice el `$top` acumulador de la `$group` fase para obtener el documento mejor clasificado por grupo según un orden de clasificación especificado.

Parámetros

- `sortBy`: un documento que especifica el orden de clasificación. 1Utilícelo para ascender o `-1` descender.

- `output`: expresión que especifica los campos que se devolverán desde la parte superior del documento.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo se muestra cómo usar el `$top` acumulador para encontrar la cantidad más vendida (cantidad más alta) por artículo de una colección de ventas.

Crea documentos de muestra

```
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 }
])
```

Ejemplo de consulta

```
db.sales.aggregate([
  { $group: { _id: "$item", topSale: { $top: { sortBy: { quantity: -1 }, output:
 { quantity: "$quantity", price: "$price" } } } } }
])
```

Salida

```
[
  { "_id": "xyz", "topSale": { "quantity": 15, "price": 3 } },
  { "_id": "abc", "topSale": { "quantity": 10, "price": 5 } }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El `$top` operador también se puede utilizar como operador de ventanas en el `$setWindowFields` escenario. En este contexto, devuelve el `output` del documento mejor clasificado (según el propio del operador`sortBy`) de entre los documentos de cada ventana. Especifique el operador en el `output` campo y, si lo desea, defina los límites de la ventana con un `window` documento.

> **Note**
>
> Cuando se usa como operador de ventana en`$setWindowFields`, `$top` está limitado a 100 MB de datos intermedios. Una operación que supera este límite devuelve un error.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

En el ejemplo siguiente se dividen los documentos`player`, se ordena cada partición y se devuelve la ronda con la puntuación más alta vista desde el principio de la partición hasta el documento actual. `round`

```
db.matchScores.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$player",
      sortBy: { round: 1 },
      output: {
        bestRoundSoFar: {
          $top: { sortBy: { score: -1 }, output: { round: "$round", score:
 "$score" } },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Salida

```
[
```

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

Cada documento se complementa con `bestRoundSoFar` la ronda con la puntuación más alta dentro de su partición hasta el documento actual, incluido.

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$top` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran tanto el uso del acumulador (en`$group`) como el uso del operador de ventanas (en`$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

Nuevo a partir de la versión 8.0.1.

Usa el `$topN` acumulador del `$group` escenario para devolver los N elementos principales de un grupo según un orden de clasificación especificado. Si un grupo contiene menos de N elementos, `$topN` devuelve todos los elementos del grupo.

Parámetros

- `n`: un entero positivo o una expresión que se resuelve en uno y que especifica el número de resultados principales que se devolverán por grupo.
- `sortBy`: un documento que especifica el orden de clasificación. 1Utilícelo para ascender o `-1` descender.
- `output`: expresión que especifica los campos que se devolverán de cada uno de los N documentos principales.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el $topN acumulador para encontrar las dos ventas más vendidas (la cantidad más alta) por artículo de una colección de ventas.

Crea documentos de muestra

```
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 }
])
```

Ejemplo de consulta

```
db.sales.aggregate([
  { $group: { _id: "$item", topTwoSales: { $topN: { n: 2, sortBy: { quantity: -1 },
 output: { quantity: "$quantity", price: "$price" } } } } }
])
```

Salida

```
[
  { "_id": "xyz", "topTwoSales": [{ "quantity": 15, "price": 3 }, { "quantity": 9,
 "price": 6 }] },
  { "_id": "abc", "topTwoSales": [{ "quantity": 10, "price": 5 }, { "quantity": 7,
 "price": 8 }] }
]
```

Ejemplo de uso de un operador de ventana (MongoDB Shell)

Nuevo a partir de la versión 8.0.2.

El $topN operador también se puede utilizar como operador de ventanas en el $setWindowFields escenario. En este contexto, devuelve una matriz output de los N documentos de cada ventana de los primeros N documentos (según los propios del operadorsortBy). Especifique el operador en el output campo y, si lo desea, defina los límites de la ventana con un window documento.

> **Note**
>
> Cuando se usa como operador de ventana en`$setWindowFields`, `$topN` está limitado a 100 MB de datos intermedios. Una operación que supera este límite devuelve un error.

Crea documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

En el ejemplo siguiente se dividen los documentos`player`, se ordena cada partición y se devuelven las dos rondas con mayor puntuación desde el principio de la partición hasta el documento actual. `round`

```
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"] }
        }
      }
    }
  }
]);
```

Salida

```
[
```

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

Cada documento se complementa con `topTwoRoundsSoFar` las dos rondas con la puntuación más alta de su partición hasta el documento actual, incluido. Si la ventana contiene menos de dos documentos, se devuelven todos.

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$topN` del operador, seleccione la pestaña correspondiente al idioma que desee utilizar. Los siguientes ejemplos muestran tanto el uso del acumulador (en`$group`) como el uso del operador de ventanas (en`$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

Nuevo desde la versión 8.0.1.

El `$trunc` operador de Amazon DocumentDB trunca un número hasta un decimal específico y elimina los dígitos sin redondear.

Parámetros

- `number`: expresión que se resuelve en un número.
- `place`: opcional. Expresión entera entre -20 y 100 que especifica el número de decimales a los que se debe truncar. Un valor negativo se trunca a la izquierda del decimal. El valor predeterminado es 0.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$trunc` operador para truncar valores numéricos a un decimal.

Truncar con un valor posicional positivo

```
db.measurements.insertMany([
  {_id: 1, value: 3.456},
  {_id: 2, value: 7.891},
  {_id: 3, value: 12.345}
]);
```

Ejemplo de consulta

```
db.measurements.aggregate([
  { $project: { truncated: { $trunc: ["$value", 1] } } }
]);
```

Salida

```
[
  {_id: 1, truncated: 3.4},
  {_id: 2, truncated: 7.8},
  {_id: 3, truncated: 12.3}
]
```

Truncar con un valor posicional negativo

Cuando `place` es negativo, `$trunc` reemplaza los dígitos situados a la izquierda del decimal por ceros. En el siguiente ejemplo, se truncan los valores con el primer dígito situado a la izquierda del decimal.

```
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] } } }
]);
```

Salida

```
[
  {_id: 1, truncated: 10},
  {_id: 2, truncated: 20},
  {_id: 3, truncated: 30}
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso `$trunc` del operador, elija la pestaña correspondiente al idioma que desee utilizar:

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 de $ t

Nuevo a partir de la versión 8.0.1.

El `$tsIncrement` operador de Amazon DocumentDB devuelve el ordinal incremental a partir de un valor de marca de tiempo como un entero largo.

Parámetros

- `expression`: una expresión que se resuelve en una marca de tiempo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$tsIncrement` operador para extraer el incremento ordinal de los valores de la marca de tiempo.

Cree documentos de muestra

```
db.events.insertMany([
  {_id: 1, ts: Timestamp(1678900000, 1)},
  {_id: 2, ts: Timestamp(1678900000, 2)},
  {_id: 3, ts: Timestamp(1678900001, 1)}
]);
```

Ejemplo de consulta

```
db.events.aggregate([
  { $project: { increment: { $tsIncrement: "$ts" } } }
]);
```

Salida

```
[
  {_id: 1, increment: Long("1")},
  {_id: 2, increment: Long("2")},
  {_id: 3, increment: Long("1")}
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$tsIncrement` operador, elige la pestaña del idioma que quieres usar:

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()
```

$tsSegundo

Nuevo desde la versión 8.0.1.

El `$tsSecond` operador de Amazon DocumentDB devuelve la parte correspondiente a los segundos de un valor de marca de tiempo como un entero largo (los segundos de la época de Unix).

Parámetros

- `expression`: una expresión que se resuelve en una marca de tiempo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$tsSecond` operador para extraer la parte de segundos de los valores de la marca de tiempo.

Cree documentos de muestra

```
db.events.insertMany([
  {_id: 1, ts: Timestamp(1678900000, 1)},
  {_id: 2, ts: Timestamp(1678900000, 2)},
  {_id: 3, ts: Timestamp(1678900001, 1)}
]);
```

Ejemplo de consulta

```
db.events.aggregate([
  { $project: { seconds: { $tsSecond: "$ts" } } }
]);
```

Salida

```
[
  {_id: 1, seconds: Long("1678900000")},
  {_id: 2, seconds: Long("1678900000")},
  {_id: 3, seconds: Long("1678900001")}
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$tsSecond` operador, elige la pestaña del idioma que quieres usar:

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

El `$toBool` operador de Amazon DocumentDB convierte una expresión en un valor booleano.

Parámetros

- `expression`: la expresión que se va a convertir en un valor booleano.

Nota: Cualquier cadena se convierte en. `true`

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra el uso del `$toBool` operador para normalizar los valores de estado del dispositivo a partir de distintos tipos de datos.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.deviceStates.aggregate([
  {
    $project: {
      _id: 1,
      deviceId: 1,
      isActive: { $toBool: "$status" }
    }
  }
]);
```

Salida

```
[
  { "_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 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$toBool` comando, elija la pestaña del idioma que desee usar:

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()
```

$toDate

Novedades de la versión 4.0

El operador de `$toDate` agregación de Amazon DocumentDB se utiliza para convertir una cadena de fecha o fecha y hora en un tipo de fecha BSON. Es la operación inversa del operador. `$dateToString`

Parámetros

- `dateString`: una representación en cadena de una fecha o fecha y hora que se va a convertir a un tipo de fecha BSON.
- `format`: (opcional) Una cadena que especifica el formato de. `dateString` Si no se proporciona, el operador intentará analizarlos `dateString` en varios formatos de fecha y hora estándar.
- `timezone`: (opcional) Una cadena que representa la zona horaria que se utilizará para la conversión. Si no se proporciona, se utiliza la zona horaria local.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$toDate` operador para convertir una cadena de fecha en un tipo de fecha BSON.

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.events.aggregate([
  {
    $project: {
      eventName: 1,
      eventTimeDate: { $toDate: "$eventTime" }
    }
  }
```

```
]);
```

Salida

```
[
  {
    "_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")
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$toDate` comando, elija la pestaña del idioma que desee usar:

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

Novedades de la versión 4.0

El `$toDecimal` operador de Amazon DocumentDB se utiliza para convertir un valor al tipo de datos Decimal128. Esto resulta útil cuando necesita realizar operaciones aritméticas decimales precisas o

manejar valores decimales grandes que no se pueden representar con precisión con el tipo de datos Double.

Parámetros

- `expression`: la expresión que se va a convertir al tipo de datos Decimal128.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo utilizar el `$toDecimal` operador para convertir un valor de cadena en un tipo de datos Decimal128.

Cree documentos de muestra

```
db.numbers.insertOne({ _id: 1, value: "3.14" });
db.numbers.insertOne({ _id: 2, value: "2.71" });
```

Ejemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    decimalValue: { $toDecimal: "$value" }
  }}
])
```

Salida

```
[
  { "_id" : 1, "decimalValue" : Decimal128("3.14") },
  { "_id" : 2, "decimalValue" : Decimal128("2.71") }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$toDecimal` comando, elija la pestaña del idioma que desee usar:

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 doble

Novedades de la versión 4.0

El `$toDouble` operador de Amazon DocumentDB se utiliza para convertir un valor en un número de punto flotante de 64 bits de doble precisión. Esto puede resultar útil cuando necesite realizar operaciones aritméticas con valores que originalmente no tenían un formato numérico.

Parámetros

`<expression>`: la expresión que se va a convertir en un valor doble. Puede ser cualquier expresión válida que se resuelva en un valor numérico, de cadena o booleano.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo utilizar el `$toDouble` operador para convertir un valor de cadena en un valor numérico con el fin de realizar un cálculo matemático.

Cree documentos de muestra

```
db.numbers.insertMany([
  { _id: 1, value: "10.5" },
  { _id: 2, value: "20.25" },
  { _id: 3, value: "7" }
])
```

Ejemplo de consulta

```
db.numbers.aggregate([
  {
    $project: {
      _id: 1,
      value: 1,
      double_value: { $toDouble: "$value" },
      double_plus_five: { $add: [{ $toDouble: "$value" }, 5] }
    }
  }
])
```

Salida

```
[
```

```
  { "_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 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$toDouble` comando, elija la pestaña del idioma que desee usar:

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()
```

$toInt

Novedades de la versión 4.0

El `$toInt` operador de Amazon DocumentDB se utiliza para convertir un valor de entrada en un tipo de datos entero. Este operador resulta útil cuando necesita asegurarse de que un campo o expresión se representa como un número entero, lo que puede ser importante para determinadas operaciones o tareas de procesamiento de datos.

Parámetros

- `expression`: la expresión que se va a convertir en un número entero.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$toInt` operador para convertir un valor de cadena en un entero.

Cree documentos de muestra

```
db.numbers.insertMany([
  { "name": "one", "value": "1" },
  { "name": "hundred", "value": "100" }
]);
```

Ejemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    "_id": 0,
    "name": 1,
    "intValue": { $toInt: "$value" }
  }}
]);
```

Salida

```
{ "name": "one", "intValue": 1 }
{ "name": "hundred", "intValue": 100 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$toInt` comando, elija la pestaña del idioma que desee usar:

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

Novedades de la versión 4.0

El `$toLong` operador de Amazon DocumentDB se utiliza para convertir un valor en un tipo de datos entero (largo) de 64 bits. Esto puede resultar útil cuando necesite realizar operaciones aritméticas o comparaciones con valores numéricos que pueden almacenarse como cadenas u otros tipos de datos.

Parámetros

- `expression`: la expresión que se va a convertir en un entero de 64 bits.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo utilizar el `$toLong` operador para convertir un valor de cadena en un entero de 64 bits.

Cree documentos de muestra

```
db.numbers.insertMany([
  { _id: 1, value: "42" },
  { _id: 3, value: "9223372036854775807" }
]);
```

Ejemplo de consulta

```
db.numbers.aggregate([
  {
    $project: {
      _id: 1,
      longValue: { $toLong: "$value" }
    }
  }
])
```

Salida

```
[
  { "_id" : 1, "longValue" : 42 },
  { "_id" : 3, "longValue" : 9223372036854775807 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$toLong` comando, elija la pestaña del idioma que desee usar:

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

El `$toLower` operador de Amazon DocumentDB se utiliza para convertir una cadena a minúsculas.

Parámetros

- `expression`: La expresión de cadena que se va a convertir a minúsculas.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$toLower` operador para convertir el `Desk` campo a minúsculas.

Cree documentos de muestra

```
db.locations.insertMany([
  { "_id": 1, "Desk": "Düsseldorf-BVV-021" },
  { "_id": 2, "Desk": "Munich-HGG-32a" }
]);
```

Ejemplo de consulta

```
db.locations.aggregate([
  { $project: { item: { $toLower: "$Desk" } } }
]);
```

Salida

```
{ "_id" : 1, "item" : "düsseldorf-bvv-021" }
{ "_id" : 2, "item" : "munich-hgg-32a" }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$toLower` comando, elija la pestaña del idioma que desee usar:

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

Nuevo desde la versión 4.0

El `$toObjectId` operador de Amazon DocumentDB se utiliza para convertir una representación de cadena de an ObjectId en un tipo de ObjectId datos real. Esto puede resultar útil cuando se trabaja con datos que se han almacenado como representaciones en cadena de caracteres ObjectIds, ya que permite realizar operaciones que requieren ese tipo de ObjectId datos.

Parámetros

- `expression`: expresión de cadena que representa un valor válido ObjectId.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$toObjectId` operador para convertir una representación de cadena de an en el tipo ObjectId de ObjectId datos.

Cree documentos de muestra

```
db.employees.insertMany([
  { _id: 1, empId:"64e5f8886218c620cf0e8f8a", name: "Carol Smith", employeeId:
 "c720a" },
  { _id: 2, empId:"64e5f94e6218c620cf0e8f8c", name: "Bill Taylor", employeeId:
 "c721a" }
]);
```

Ejemplo de consulta

```
db.employees.aggregate([
  { $project: {
    "empIdAsObjectId": {$toObjectId: "$empId"}}
  }
]);
```

Salida

```
[
  { _id: 1, empIdAsObjectId: ObjectId('64e5f8886218c620cf0e8f8a') },
  { _id: 2, empIdAsObjectId: ObjectId('64e5f94e6218c620cf0e8f8c') }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$toObjectId` comando, elija la pestaña del idioma que desee usar:

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

Novedades de la versión 4.0

El `$toString` operador de Amazon DocumentDB se utiliza para convertir un valor de cualquier tipo (excepto nulo) en una representación de cadena. Esto puede resultar útil cuando necesite realizar operaciones de cadena en valores que originalmente no están en formato de cadena.

Parámetros

- `expression`: la expresión que se va a convertir en cadena.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$toString` operador para convertir valores numéricos en cadenas.

Cree documentos de muestra

```
db.numbers.insertMany([
  { "_id": 1, "value": 42 },
  { "_id": 2, "value": 3.14 }
]);
```

Ejemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    valueAsString: { $toString: "$value" }
  }}
]);
```

Salida

```
{ "_id": 1, "valueAsString": "42" }
{ "_id": 2, "valueAsString": "3.14" }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$toString` comando, elija la pestaña del idioma que desee usar:

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

El `$toUpper` operador de Amazon DocumentDB se utiliza para convertir una cadena a mayúsculas.

Parámetros

- `expression`: La expresión de cadena que se va a convertir a mayúsculas.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$toUpper` operador para convertir el `Desk` campo a mayúsculas.

Cree documentos de muestra

```
db.locations.insertMany([
  { "_id": 1, "Desk": "düsseldorf-bvv-021" },
  { "_id": 2, "Desk": "munich-hgg-32a" }
]);
```

Ejemplo de consulta

```
db.locations.aggregate([
  { $project: { item: { $toUpper: "$Desk" } } }
]);
```

Salida

```
{ "_id" : 1, "item" : "DüSSELDORF-BVV-021" }
{ "_id" : 2, "item" : "MUNICH-HGG-32A" }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$toUpper` comando, elija la pestaña del idioma que desee usar:

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()
```

$ a UUID

Nuevo a partir de la versión 8.0.1.

El `$toUUID` operador de Amazon DocumentDB convierte un valor de cadena en un UUID (subtipo binario 4).

Parámetros

- `expression`: una expresión que se resuelve en una cadena en formato UUID (por ejemplo, «12345678-1234-1234-123456789abc»).

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$toUUID` operador para convertir valores de cadena al formato UUID.

Cree documentos de muestra

```
db.records.insertMany([
  {_id: 1, uuidStr: "550e8400-e29b-41d4-a716-446655440000"},
  {_id: 2, uuidStr: "6ba7b810-9dad-11d1-80b4-00c04fd430c8"}
]);
```

Ejemplo de consulta

```
db.records.aggregate([
  { $project: { uuid: { $toUUID: "$uuidStr" } } }
]);
```

Salida

```
[
  {_id: 1, uuid: UUID("550e8400-e29b-41d4-a716-446655440000")},
  {_id: 2, uuid: UUID("6ba7b810-9dad-11d1-80b4-00c04fd430c8")}
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$toUUID` operador, elige la pestaña del idioma que quieres usar:

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

Nuevo a partir de la versión 4.0

El `$trim` operador de Amazon DocumentDB se utiliza para eliminar los espacios en blanco and/or iniciales de una cadena.

Parámetros

- `input`: la expresión de cadena que se va a recortar.
- `chars`: (opcional) Especifica los caracteres que se van a recortar desde el principio y el final de la entrada; el valor predeterminado es el espacio en blanco.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$trim` operador para eliminar los espacios en blanco iniciales y finales de una cadena.

Cree documentos de muestra

```
db.people.insertMany([
  { "name": "   John Doe   " },
  { "name": "   Bob Johnson   " }
])
```

Ejemplo de consulta

```
db.people.aggregate([
  { $project: {
    "name": { $trim: {input: "$name"}}
```

```
  }}
])
```

Salida

```
[
  { "name": "John Doe" },
  { "name": "Bob Johnson" }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$trim` comando, elige la pestaña del idioma que quieres usar:

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

El operador de `$type` agregación devuelve el tipo de datos BSON de un campo especificado. Esto resulta útil para identificar el tipo de datos de los valores de campo durante las operaciones de agregación.

Parámetros

- `expression`: el campo o la expresión cuyo tipo se va a devolver.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra el uso `$type` del operador para identificar el tipo de datos del campo de precio de cada producto.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.inventory.aggregate([
  {
    $project: {
      item: 1,
      price: 1,
      priceType: { $type: "$price" }
    }
  }
]);
```

Salida

```
[
  { _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' }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el operador de `$type` agregación, elija la pestaña correspondiente al idioma que desee usar:

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

Novedades de la versión 8.0

No es compatible con el clúster de Elastic.

La etapa de `$unset` agregación de Amazon DocumentDB le permite eliminar campos de los documentos.

Parámetros

- `expression`: nombre de campo o lista de varios nombres de campo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso de la etapa de `$unset` agregación para eliminar el `price` campo.

Cree documentos de muestra

```
db.inventory.insertMany([
  { item: "pencil", quantity: 100, price: 0.24},
  { item: "pen", quantity: 204, price: 1.78 }
]);
```

Ejemplo de agregación

```
db.inventory.aggregate([
  {
    $unset: "price"
  }
])
```

Salida

```
[
  {
    _id: ObjectId('69248951d66dcae121d2950d'),
    item: 'pencil',
    quantity: 100
  },
  {
    _id: ObjectId('69248951d66dcae121d2950e'),
    item: 'pen',
    quantity: 204
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$unset` comando, elija la pestaña del idioma que desee usar:

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

El `$unwind` operador se utiliza para deconstruir un campo de matriz a partir de los documentos de entrada para generar un documento para cada elemento. Esto puede resultar útil cuando desee realizar operaciones en los elementos individuales de una matriz, como filtrar, ordenar o transformar los datos.

Parámetros

- `path`: La ruta al campo de matriz que se va a desenrollar.
- `includeArrayIndex`: (opcional) Especifica el nombre del nuevo campo que contiene el índice del elemento de la matriz.
- `preserveNullAndEmptyArrays`: (opcional) Determina si la operación conserva el documento original cuando el campo de matriz es nulo o es una matriz vacía.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$unwind` operador para deconstruir un campo de matriz y realizar más operaciones en los elementos individuales.

Cree documentos de muestra

```
db.people.insertMany([
  { _id: 1, name: "jon", hobbies: ["painting", "dancing", "singing"] },
  { _id: 2, name: "jane", hobbies: ["reading", "swimming"] },
  { _id: 3, name: "jack", hobbies: [] }
])
```

Ejemplo de consulta

```
db.people.aggregate([
  { $unwind: "$hobbies" }
```

```
])
```

Salida

```
[
  { _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' }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$unwind` comando, elija la pestaña del idioma que desee usar:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('people');

  const result = await collection.aggregate([
    { $unwind: '$hobbies' }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient
```

```
def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['people']

    result = list(collection.aggregate([
        { '$unwind': '$hobbies' }
    ]))

    print(result)
    client.close()

example()
```

$ VectorSearch

Nuevo desde la versión 8.0

No es compatible con el clúster de Elastic.

El `$vectorSearch` operador de Amazon DocumentDB le permite realizar una búsqueda vectorial, un método utilizado en el aprendizaje automático para encontrar puntos de datos similares mediante la comparación de sus representaciones vectoriales mediante métricas de distancia o similitud. Esta capacidad combina la flexibilidad y las sofisticadas consultas de una base de datos de JSON-based documentos con la potencia de la búsqueda vectorial, lo que le permite desarrollar casos de uso del aprendizaje automático y de la IA generativa, como la búsqueda semántica, la recomendación de productos, etc.

Parámetros

- `<exact>`(opcional): marca que especifica si se debe ejecutar la búsqueda del vecino más cercano exacto (ENN) o del vecino más cercano aproximado (ANN). El valor puede ser uno de los siguientes:
- false: para ejecutar una búsqueda ANN
- true: para ejecutar la búsqueda ENN

Si se omite o se establece en falso, `numCandidates` es obligatorio.

```
- `<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.
```

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$vectorSearch` operador para buscar descripciones de productos similares en función de sus representaciones vectoriales.

Cree documentos de muestra

```
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 índice de búsqueda vectorial

```
db.runCommand(
    {
        createIndexes: "products",
```

```
        indexes: [{
            key: {
                "description_vector": "vector"
            },
            vectorOptions: {
                type: "hnsw",
                dimensions: 3,
                similarity: "cosine",
                m: 16,
                efConstruction: 64
            },
            name: "description_index"
        }]
    }
);
```

Ejemplo de consulta

```
db.products.aggregate([
  { $vectorSearch: {
      index: "description_index",
      limit: 2,
      numCandidates: 10,
      path: "description_vector",
      queryVector: [0.1, 0.2, 0.3]
    }
  }
]);
```

Salida

```
[
  {
    "_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 ]
```

```
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$vectorSearch` comando, elija la pestaña del idioma que desee usar:

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

El `$week` operador de Amazon DocumentDB devuelve el número de semana de una fecha (0-53) según la norma ISO 8601. El número de semana se calcula en función del año y el día de la semana, y el lunes es el primer día de la semana.

Parámetros

Ninguno

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$week` operador para recuperar el número de semana de una fecha determinada.

Cree documentos de muestra

```
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") }
]);
```

Ejemplo de consulta

```
db.events.aggregate([
  { $project: {
    _id: 1,
    week: { $week: "$date" }
  }}
]);
```

Salida

```
[
  { "_id": 1, "week": 1 },
  { "_id": 2, "week": 2 },
  { "_id": 3, "week": 53 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el $week comando, elija la pestaña del idioma que desee usar:

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

El `$year` operador de Amazon DocumentDB extrae el componente del año de una fecha o marca horaria.

Parámetros

- `expression`: La expresión de fecha o marca horaria de la que se extrae el componente del año.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$year` operador para extraer el componente de año de un campo de fecha.

Cree documentos de muestra

```
db.events.insertMany([
  { "_id": 1, "date": ISODate("2023-04-15T00:00:00Z") },
  { "_id": 3, "date": ISODate("2021-12-31T00:00:00Z") }
]);
```

Ejemplo de consulta

```
db.events.aggregate([
  { $project: { year: { $year: "$date" } } }
]);
```

Salida

```
[
  { "_id": 1, "year": 2023 },
  { "_id": 3, "year": 2021 }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$year` comando, elija la pestaña del idioma que desee usar:

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

El `$zip` operador de Amazon DocumentDB le permite combinar varias matrices en una sola matriz de tuplas (pares clave-valor). Esto resulta útil cuando necesita crear nuevos documentos u objetos mediante la combinación de datos de distintas fuentes o matrices dentro de un documento.

Parámetros

- `inputs`: conjunto de expresiones que se convierten en matrices. Estas matrices se combinarán en una sola matriz de tuplas.

- `useLongestLength`: (opcional) Si`true`, la matriz de salida tendrá la longitud de la matriz de entrada más larga, rellenando las matrices más cortas con valores. `null` Si`false`, la matriz de salida tendrá la longitud de la matriz de entrada más corta.
- `defaults`: (opcional) Una matriz de valores predeterminados para usar en las tuplas si la matriz de entrada correspondiente es más corta que la matriz de entrada más larga y lo `useLongestLength` es`true`.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$zip` operador para combinar dos matrices en una sola matriz de tuplas.

Cree documentos de muestra

```
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"]
  }
])
```

Ejemplo de consulta

```
db.grades.aggregate([
  {
    $project: {
      "name": 1,
      "scoredClasses": {
        $zip: {
          inputs: ["$scores", "$classes"],
          useLongestLength: true,
          defaults: [null, null]
        }
```

```
      }
    }
  }
])
```

Salida

```
[
  {
    "_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"]
    ]
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$zip` comando, elija la pestaña del idioma que desee usar:

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()
```

Geospatial (Geoespacial)

En esta sección se proporciona información detallada sobre los operadores geoespaciales compatibles con Amazon DocumentDB.

Temas

- $geometry
- $geoIntersects
- $geoWithin
- $maxDistance
- $minDistance
- $near
- $nearSphere

$geometry

El `$geometry` operador de Amazon DocumentDB se utiliza para especificar un objeto geométrico de GeoJSON como parte de una consulta geoespacial. Este operador se utiliza junto con otros operadores de consultas geoespaciales, por ejemplo`$geoWithin`, `$geoIntersects` para realizar consultas espaciales en sus datos.

En Amazon DocumentDB, el `$geometry` operador admite los siguientes tipos de geometría de GeoJSON:

- Point

- LineString
- Polygon
- MultiPoint
- MultiLineString
- MultiPolygon
- GeometryCollection

Parámetros

- `type`: El tipo del objeto geométrico de GeoJSON, por ejemplo,, `PointPolygon`, etc.
- `coordinates`: Una matriz de coordenadas que representa la geometría. La estructura de la matriz de coordenadas depende del tipo de geometría.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra cómo utilizar el `$geometry` operador para realizar una `$geoIntersects` consulta en Amazon DocumentDB.

Cree documentos de muestra

```
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]
      ]]
    }
  }
]);
```

Ejemplo de consulta

```
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]
        ]]
      }
    }
  }
})
```

Salida

```
[
  {
    "_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]
        ]
      ]
```

```
    }
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$geometry` comando, elija la pestaña del idioma que desee usar:

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

El `$geoIntersects` operador de Amazon DocumentDB se utiliza para buscar documentos cuyos datos geoespaciales se cruzan con un objeto GeoJSON específico. Este operador es útil para

aplicaciones que requieren identificar documentos en función de su relación espacial con una forma geográfica determinada, como un polígono o un multipolígono.

Parámetros

- `$geometry`: un objeto GeoJSON que representa la forma para comprobar la intersección. Los tipos de objetos GeoJSON compatibles son `Point` `LineStringPolygon`, y. `MultiPolygon`

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra cómo utilizar el `$geoIntersects` operador para buscar el nombre de estado de un conjunto de coordenadas determinado en Amazon DocumentDB.

Cree documentos de muestra

```
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]
      ]]
    }
  }
```

```
]);
```

Ejemplo de consulta

```
var location = [-73.965355, 40.782865];

db.states.find({
  "loc": {
    "$geoIntersects": {
      "$geometry": {
        "type": "Point",
        "coordinates": location
      }
    }
  }
}, {
  "name": 1
});
```

Salida

```
{ "_id" : ObjectId("536b0a143004b15885c91a2c"), "name" : "New York" }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$geoIntersects` comando, elija la pestaña del idioma que desee usar:

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

El `$geoWithin` operador de Amazon DocumentDB se utiliza para buscar documentos cuyos datos de ubicación (representados como objetos GeoJSON) estén completamente contenidos en una forma específica, como un polígono o un multipolígono. Esto resulta útil para consultar objetos que se encuentran dentro de una región geográfica específica.

Parámetros

- `$geometry`: un objeto GeoJSON que representa la forma sobre la que se va a realizar la consulta.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$geoWithin` operador para buscar todos los aeropuertos ubicados en el estado de Nueva York.

Cree documentos de muestra

```
// 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]
        }
```

```
    }
]);
```

Ejemplo de consulta

```
var state = db.states.findOne({"name": "New York"});

db.airports.find({
    "loc": {
        "$geoWithin": {
            "$geometry": state.loc
        }
    }
}, {
    "name": 1,
    "type": 1,
    "code": 1,
    "_id": 0
});
```

Salida

```
[
  {
    "name": "John F. Kennedy International Airport",
    "type": "airport",
    "code": "JFK"
  },
  {
    "name": "LaGuardia Airport",
    "type": "airport",
    "code": "LGA"
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$geoWithin` comando, elija la pestaña del idioma que desee usar:

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

El `$maxDistance` operador de Amazon DocumentDB se utiliza para especificar la distancia máxima (en metros) desde un punto GeoJSON a la que deben estar los documentos para incluirlos en los resultados de la consulta. Este operador se usa junto con el `$nearSphere` operador para realizar consultas geoespaciales.

Parámetros

- `$maxDistance`: La distancia máxima (en metros) desde el punto de referencia a la que deben estar los documentos para incluirlos en los resultados de la consulta.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra cómo utilizar el `$maxDistance` operador en Amazon DocumentDB para buscar todas las capitales de estado en un radio de 100 kilómetros de Boston.

Cree documentos de muestra

```
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] } }
```

```
]);
```

Ejemplo de consulta

```
db.capitals.find(
  {
    location: {
      $nearSphere: {
        $geometry: { type: "Point", coordinates: [-71.0589, 42.3601] },
        $maxDistance: 100000
      }
    }
  },
  { state: 1, city: 1, _id: 0 }
);
```

Salida

```
[
  { "state": "Rhode Island", "city": "Providence" },
  { "state": "New Hampshire", "city": "Concord" }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$maxDistance` comando, elija la pestaña del idioma que desee usar:

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

`$minDistance`es un operador de búsqueda que se utiliza junto con `$nearSphere` o `$geoNear` para filtrar documentos que se encuentran al menos a la distancia mínima especificada desde el punto central. Este operador es compatible con Amazon DocumentDB y funciona de forma similar a su homólogo en MongoDB.

Parámetros

- `$minDistance`: La distancia mínima (en metros) desde el punto central para incluir documentos en los resultados.

Ejemplo (MongoDB Shell)

En este ejemplo, buscaremos todos los restaurantes en un radio de 2 kilómetros de una ubicación específica en Seattle, Washington.

Cree documentos de muestra

```
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]
    }
  }
]);
```

Ejemplo de consulta

```
db.usarestaurants.find({
  "location": {
    "$nearSphere": {
      "$geometry": {
        "type": "Point",
        "coordinates": [-122.3516, 47.6156]
      },
      "$minDistance": 1,
      "$maxDistance": 2000
    }
  }
}, {
  "name": 1
});
```

Salida

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
{ "_id" : ObjectId("611f3da985009a81ad38e74c"), "name" : "Pike Place Grill" }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$minDistance` comando, elija la pestaña del idioma que desee usar:

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

El `$near` operador de Amazon DocumentDB se utiliza para buscar documentos que se encuentran geográficamente cerca de un punto específico. Devuelve los documentos ordenados por distancia, con los documentos más cercanos primero. Este operador requiere un índice geoespacial de 2dsphere y es útil para consultas de proximidad sobre datos de ubicación.

Parámetros

- `$geometry`: un objeto GeoJSON Point que define el punto central de la consulta de proximidad.
- `$maxDistance`: (opcional) La distancia máxima en metros desde el punto especificado que puede ocupar un documento para que coincida con la consulta.
- `$minDistance`: (opcional) La distancia mínima en metros desde el punto especificado en la que puede estar un documento para que coincida con la consulta.

Requisitos del índice

- `2dsphere index`: Necesario para consultas geoespaciales sobre datos de puntos de GeoJSON.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$near` operador para buscar los restaurantes más cercanos a una ubicación específica en Seattle, Washington.

Cree documentos de muestra

```
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] }
  }
]);
```

Cree el índice de 2dsphere

```
db.usarestaurants.createIndex({ "location": "2dsphere" });
```

Ejemplo de consulta con GeoJSON Point

```
db.usarestaurants.find({
  location: {
    $near: {
      $geometry: {
        type: "Point",
        coordinates: [-122.3516, 47.6156]
      },
      $maxDistance: 100,
      $minDistance: 10
    }
```

```
  }
});
```

Salida

```
{
  "_id" : ObjectId("69031ec9ea1c2922a1ce5f4a"),
  "name" : "Noodle House",
  "city" : "Seattle",
  "state" : "Washington",
  "rating" : 4.8,
  "location" : {
    "type" : "Point",
    "coordinates" : [ -122.3517, 47.6159 ]
  }
}
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$near` comando, elija la pestaña del idioma que desee usar:

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

El `$nearSphere` operador de Amazon DocumentDB se utiliza para buscar documentos que se encuentran a una distancia específica de un punto geoespacial. Este operador es particularmente útil para consultas geoespaciales, como encontrar todos los restaurantes dentro de un radio determinado de una ubicación determinada.

Parámetros

- `$geometry`: un objeto GeoJSON que representa el punto de referencia. Debe ser un `Point` objeto con `coordinates` campos `type` y.
- `$minDistance`: (opcional) La distancia mínima (en metros) desde el punto de referencia que deben estar los documentos.
- `$maxDistance`: (opcional) La distancia máxima (en metros) desde el punto de referencia que deben estar los documentos.

Ejemplo (MongoDB Shell)

En este ejemplo, encontraremos todos los restaurantes en un radio de 2 kilómetros (2000 metros) de una ubicación específica en Seattle, Washington.

Cree documentos de muestra

```
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] }
  }
```

```
]);
```

Ejemplo de consulta

```
db.usarestaurants.find({
  location: {
    $nearSphere: {
      $geometry: {
        type: "Point",
        coordinates: [-122.3516, 47.6156]
      },
      $minDistance: 1,
      $maxDistance: 2000
    }
  }
}, {
  name: 1
});
```

Salida

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
{ "_id" : ObjectId("611f3da985009a81ad38e74c"), "name" : "Pike Place Grill" }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$nearSphere` comando, elija la pestaña del idioma que desee usar:

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()
```

Operadores de consulta y proyección

Esta sección proporciona información detallada sobre los operadores de consulta y proyección compatibles con Amazon DocumentDB.

Temas

- $
- $all
- $and
- $ bits AllClear
- $ bits AllSet
- $ bits AnyClear
- $ bits AnySet
- $comment
- $elemMatch
- $eq
- $exists
- $expr
- $gt
- $gte
- $in
- $jsonSchema
- $lt
- $mod
- $lte
- $meta

- $ne
- $nin
- $nor
- $not
- $or
- $regex
- $sampleRate
- $slice
- $size
- $text
- $type

$

El operador de $ proyección limita el contenido de un campo de matriz para que devuelva solo el primer elemento que coincida con la condición de consulta. Se utiliza para proyectar un único elemento de matriz coincidente.

Parámetros

- `field.$`: El campo de matriz con el operador posicional para proyectar el primer elemento coincidente.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del operador de $ proyección para devolver solo el elemento de matriz coincidente.

Cree documentos de muestra

```
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] }
]);
```

Ejemplo de consulta

```
db.students.find(
  { grades: { $gte: 90 } },
  { name: 1, "grades.$": 1 }
);
```

Salida

```
{ "_id" : 1, "name" : "Alice", "grades" : [ 92 ] }
{ "_id" : 2, "name" : "Bob", "grades" : [ 92 ] }
{ "_id" : 3, "name" : "Charlie", "grades" : [ 95 ] }
```

En este ejemplo, solo se devuelve la primera calificación que sea mayor o igual a 90 para cada alumno.

Ejemplos de código

Para ver un ejemplo de código para usar el operador de $ proyección, elija la pestaña correspondiente al idioma que desee usar:

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

El `$all` operador de Amazon DocumentDB se usa para hacer coincidir los documentos en los que el valor de un campo es una matriz y contiene todos los elementos especificados, independientemente del orden de los elementos de la matriz.

Parámetros

- `field`: el nombre del campo que se va a comprobar.
- `[value1, value2, ...]`: la lista de valores que deben coincidir en la matriz.

Utilización **`$elemMatch`** dentro de una **`$all`** expresión

Consulte Usar `$elemMatch` dentro de una expresión `$all` para conocer las limitaciones relacionadas con el uso del `$elemMatch` operador en una `$all` expresión.

Dólares ($) en los nombres de campo

Consulte Dólar($) y punto(.) en los nombres de campos para conocer las limitaciones relacionadas con la consulta de campos con $ prefijo `$all` en objetos anidados.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$all` operador para recuperar documentos en los que el campo «Colores» es una matriz que contiene tanto «rojo» como «azul».

Cree documentos de muestra

```
db.example.insertMany([
  { "Item": "Pen", "Colors": ["Red", "Blue", "Green"] },
  { "Item": "Notebook", "Colors": ["Blue", "White"] },
  { "Item": "Poster Paint", "Colors": ["Red", "Yellow", "White"] }
])
```

Ejemplo de consulta

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

Salida

```
{
  "_id" : ObjectId("6137d6c5b3a1d35e0b6ee6ad"),
  "Item" : "Pen",
  "Colors" : [
          "Red",
          "Blue",
          "Green"
  ]
}
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$all` comando, elige la pestaña del idioma que quieres usar:

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

El `$and` operador de Amazon DocumentDB se usa para combinar varias expresiones y evaluarlas como una sola condición. Devuelve `true` si todas las expresiones proporcionadas se evalúan como y si `false` no. `true` Este operador es útil para aplicar varios criterios a una consulta.

Parámetros

- `expression1`: expresión obligatoria que se evalúa como un valor booleano.
- `expression2`: expresión obligatoria que se evalúa como un valor booleano.
- `...`: expresiones obligatorias adicionales que dan como resultado valores booleanos.

Ejemplo (MongoDB Shell)

En el ejemplo siguiente se muestra el uso del `$and` operador para buscar todos los documentos de la colección «usuarios» en los que el campo «edad» es mayor de 18 años y el campo «estado» es «activo».

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.users.find({
  $and: [
    { age: { $gt: 18 } },
    { status: "active" }
  ]
});
```

Salida

```
[
  { "_id" : ObjectId("614e3c4b63f5892e7c4e2345"), "name" : "John", "age" : 25,
 "status" : "active" },
  { "_id" : ObjectId("614e3c4b63f5892e7c4e2347"), "name" : "Alice", "age" : 22,
 "status" : "active" }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del $and comando, elige la pestaña del idioma que quieres usar:

Node.js

```
const { MongoClient } = require('mongodb');

async function findActiveUsersOlderThan18() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const users = db.collection('users');

  const activeUsersOlderThan18 = await users.find({
    $and: [
      { age: { $gt: 18 } },
      { status: 'active' }
    ]
  }).toArray();

  console.log(activeUsersOlderThan18);
  await client.close();
}

findActiveUsersOlderThan18();
```

Python

```
from pymongo import MongoClient

def find_active_users_older_than_18():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    users = db['users']

    active_users_older_than_18 = list(users.find({
        '$and': [
            {'age': {'$gt': 18}},
```

```
            {'status': 'active'}
        ]
    }))

    print(active_users_older_than_18)
    client.close()

find_active_users_older_than_18()
```

$ bits AllClear

El `$bitsAllClear` operador de Amazon DocumentDB se utiliza para hacer coincidir los documentos en los que se borran todos los bits especificados de un campo (establecido en 0). Este operador puede resultar útil para realizar operaciones bit a bit en los datos almacenados.

Parámetros

- `field`: El campo para comprobar si se están borrando los bits especificados.
- `value`: la máscara de bits numérica que especifica los bits que se deben comprobar o una lista de las posiciones de los bits que se van a comprobar. Una máscara de bits numérica puede tener una forma binaria (0b...), decimal, hexadecimal (0x...), octal (0o...) o binaria (). BinData En una lista de posiciones de bits, la posición del bit menos significativo es 0.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra el uso del `$bitsAllClear` operador en Amazon DocumentDB.

Cree documentos de muestra

```
db.collection.insertMany([
  { _id: 1, bits: 0b1010 },
  { _id: 2, bits: 0b1100 },
  { _id: 3, bits: 0b0101 }
]);
```

Ejemplo de consulta

```
db.collection.find({
  bits: { $bitsAllClear: 0b0011 }
```

```
})
```

Salida

```
{ "_id" : 2, "bits" : 12 }
```

La consulta comprueba si todos los bits especificados por la máscara de bits `0b0011` (los dos bits menos significativos) están borrados del `bits` campo. El documento con `_id` 2 cumple esta condición, ya que su `bits` campo tiene esos bits borrados.

Ejemplos de código

Para ver un ejemplo de código para usar el `$bitsAllClear` comando, elija la pestaña correspondiente al idioma que desee usar:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('collection');

  const result = await collection.find({
    bits: { $bitsAllClear: 0b0011 }
  }).toArray();

  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
```

```
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['collection']

    result = list(collection.find({
        'bits': { '$bitsAllClear': 0b0011 }
    }))

    print(result)

    client.close()

example()
```

$ bits AllSet

El `$bitsAllSet` operador de Amazon DocumentDB se utiliza para consultar documentos en los que un conjunto específico de bits está establecido en 1 en un campo. Este operador le permite realizar operaciones bit a bit en valores de campo y puede resultar útil cuando necesita comprobar el estado de los bits individuales dentro de un valor numérico.

Parámetros

- `field`: el nombre del campo numérico en el que se va a realizar la operación bit a bit.
- `value`: la máscara de bits numérica que especifica los bits que se deben comprobar o una lista de las posiciones de los bits que se van a comprobar. Una máscara de bits numérica puede tener una forma binaria (0b...), decimal, hexadecimal (0x...), octal (0o...) o binaria (). BinData En una lista de posiciones de bits, la posición del bit menos significativo es 0.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra cómo utilizar el `$bitsAllSet` operador para buscar documentos en los que el `flags` campo tenga todos los bits especificados por la máscara de bits.

Cree documentos de muestra

```
db.collection.insert([
  { _id: 1, flags: 0b1010 },
```

```
  { _id: 2, flags: 0b1100 },
  { _id: 3, flags: 0b1110 }
])
```

Ejemplo de consulta

```
db.collection.find({ flags: { $bitsAllSet: 0b1100 } })
```

Salida

```
{ "_id": 2, "flags": 12 },
{ "_id": 3, "flags": 14 }
```

En este ejemplo, la consulta busca documentos en los que el `flags` campo tenga establecidos todos los bits especificados por la máscara de bits `0b1100` (que representa el valor decimal 12). Los documentos con `_id` 2 y 3 cumplen este criterio, ya que sus valores de `flags` campo tienen configurados todos los bits necesarios (el tercero y el cuarto bits menos significativos).

Ejemplos de código

Para ver un ejemplo de código para usar el `$bitsAllSet` comando, elija la pestaña correspondiente al idioma que desee usar:

Node.js

```
const { MongoClient } = require('mongodb');

async function findWithBitsAllSet() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('collection');

  const result = await collection.find({ flags: { $bitsAllSet:
 0b1100 } }).toArray();
  console.log(result);

  await client.close();
}
```

```
findWithBitsAllSet();
```

Python

```
from pymongo import MongoClient

def find_with_bits_all_set():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.collection

    result = list(collection.find({ 'flags': { '$bitsAllSet': 0b1100 } }))
    print(result)

    client.close()

find_with_bits_all_set()
```

$ bits AnyClear

El `$bitsAnyClear` operador de Amazon DocumentDB se utiliza para hacer coincidir los documentos en los que se borra alguno de los bits especificados de un campo (establecido en 0). Esto puede resultar útil para realizar operaciones bit a bit en valores de campo almacenados en documentos.

Parámetros

- `field`: El campo que se va a comprobar.
- `value`: la máscara de bits numérica que especifica los bits que se deben comprobar o una lista de las posiciones de los bits que se van a comprobar. Una máscara de bits numérica puede tener una forma binaria (0b...), decimal, hexadecimal (0x...), octal (0o...) o binaria (). BinData En una lista de posiciones de bits, la posición del bit menos significativo es 0.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$bitsAnyClear` operador para comprobar si algún bit está despejado en el `status` campo de la `items` colección.

Cree documentos de muestra

```
db.items.insertMany([
  { "_id": 1, "status": 7 },
  { "_id": 2, "status": 15 },
  { "_id": 3, "status": 31 }
]);
```

Ejemplo de consulta

```
db.items.find({ "status": { $bitsAnyClear: 8 } })
```

Salida

```
{ "_id" : 1, "status" : 7 }
```

En este ejemplo, la consulta busca documentos en los que el `status` campo tenga algún bit libre (0) en la máscara de bits 8 (binario`1000`). El documento con `status` valores de 7 (binario111) coincide con la consulta, ya que tiene al menos un bit libre en la máscara de bits proporcionada. El bit transparente coincidente es el cuarto bit menos significativo.

Ejemplos de código

Para ver un ejemplo de código para usar el `$bitsAnyClear` comando, elija la pestaña correspondiente al idioma que desee usar:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('items');

  const result = await collection.find({ "status": { $bitsAnyClear:
 8 } }).toArray();
  console.log(result);
```

```
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.items

    result = list(collection.find({ "status": { "$bitsAnyClear": 8 } }))
    print(result)

    client.close()

example()
```

$ bits AnySet

El `$bitsAnySet` operador de Amazon DocumentDB se utiliza para consultar documentos en los que al menos un bit está establecido en 1 en los bits especificados de un campo. Este operador le permite realizar operaciones bit a bit en los valores almacenados en los campos, lo que permite realizar consultas y análisis eficientes de datos con características bit a bit.

Parámetros

- `field`: el nombre del campo al que se va a aplicar la operación bit a bit.
- `value`: la máscara de bits numérica que especifica los bits que se deben comprobar o una lista de las posiciones de los bits que se van a comprobar. Una máscara de bits numérica puede tener una forma binaria (0b...), decimal, hexadecimal (0x...), octal (0o...) o binaria (). BinData En una lista de posiciones de bits, la posición del bit menos significativo es 0.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$bitsAnySet` operador para buscar documentos en los que haya al menos un bit establecido en el `flags` campo.

Cree documentos de muestra

```
db.collection.insertMany([
  { _id: 1, flags: 0b1010 },
  { _id: 2, flags: 0b1100 },
  { _id: 3, flags: 0b0011 },
  { _id: 4, flags: 0b0100 }
]);
```

Ejemplo de consulta

```
db.collection.find({
  flags: { $bitsAnySet: 0b1010 }
})
```

Salida

```
{ "_id" : 1, "flags" : 10 }
{ "_id" : 2, "flags" : 12 }
{ "_id" : 3, "flags" : 3 }
```

La consulta devuelve los documentos en los que al menos uno de los bits especificados en la máscara de bits `0b1010` está establecido en el `flags` campo.

Ejemplos de código

Para ver un ejemplo de código para usar el `$bitsAnySet` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$comment` operador de Amazon DocumentDB se utiliza para adjuntar un comentario a una consulta. Esto puede resultar útil para proporcionar contexto o información adicional sobre la consulta, lo que puede resultar útil para la depuración o la documentación. El comentario adjunto aparecerá como parte del resultado de operaciones como db.currentOp ().

Parámetros

- `string`: El comentario adjunto a la consulta.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra cómo utilizar el `$comment` operador en Amazon DocumentDB.

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.users.find({ age: { $gt: 25 } }, { _id: 0, name: 1, age: 1 }).comment("Retrieve
 users older than 25");
```

Salida

```
{ "name" : "John Doe", "age" : 30 }
{ "name" : "Bob Johnson", "age" : 35 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$comment` comando, elija la pestaña del idioma que desee usar:

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

El `$elemMatch` operador de Amazon DocumentDB se utiliza para consultar un campo de matriz y devolver documentos en los que al menos un elemento de la matriz coincide con los criterios especificados. Este operador resulta especialmente útil cuando tiene estructuras de datos complejas con matrices anidadas o documentos incrustados.

La versión 2.0 de Planner agregó compatibilidad con índices para. `$elemMatch`

Parámetros

- `field`: El campo de matriz que se va a consultar.

- `query`: Los criterios que deben compararse con los elementos de la matriz.

Utilización **`$elemMatch`** dentro de una **`$all`** expresión

Consulte Usar `$elemMatch` dentro de una expresión `$all` las limitaciones relacionadas con el uso del `$elemMatch` operador en una `$all` expresión.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$elemMatch` operador para buscar documentos en los que la `parts` matriz tenga al menos un elemento que coincida con los criterios especificados.

Cree documentos de muestra

```
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 }] }
]);
```

Ejemplo de consulta

```
db.col.find({
  parts: { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } }
})
```

Salida

```
{ "_id" : 1, "parts" : [ { "part" : "xyz", "qty" : 10 }, { "part" : "abc", "qty" :
 20 } ] }
{ "_id" : 2, "parts" : [ { "part" : "xyz", "qty" : 5 }, { "part" : "abc", "qty" :
 10 } ] }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$elemMatch` comando, elija la pestaña del idioma que desee usar:

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

El `$eq` operador de Amazon DocumentDB se utiliza para hacer coincidir documentos en los que el valor de un campo es igual al valor especificado. Este operador se utiliza habitualmente en el `find()` método para recuperar documentos que cumplen los criterios especificados.

Parámetros

- `field`: el campo para comprobar la condición de igualdad.
- `value`: el valor que se va a comparar con el campo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$eq` operador para buscar todos los documentos en los que el `name` campo es igual a. `"Thai Curry Palace"`

Cree documentos de muestra

```
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"] }
]);
```

Ejemplo de consulta

```
db.restaurants.find({ name: { $eq: "Thai Curry Palace" } });
```

Salida

```
{ "_id" : ObjectId("68ee586f916df9d39f3d9414"), "name" : "Thai Curry Palace",
 "cuisine" : "Thai", "features" : [ "Private Dining" ] }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$eq` comando, elija la pestaña del idioma que desee usar:

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

El `$exists` operador se utiliza para comprobar si un campo existe en un documento o no. `$exists`resulta especialmente útil cuando se trabaja con índices dispersos en Amazon DocumentDB, donde el campo indexado puede no estar presente en todos los documentos.

Para utilizar un índice disperso que haya creado en una consulta, debe utilizar la cláusula `$exists` en los campos incluidos en el índice. Si lo omite`$exists`, Amazon DocumentDB no utilizará el índice disperso para la consulta.

Parámetros

- `field`: el nombre del campo para comprobar su existencia.
- `value`: un valor booleano (`true`o`false`) que especifica si el campo debe existir (`true`) o no existir (`false`) en los documentos coincidentes.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$exists` operador con un índice disperso en el `special_diets` campo de la colección. `food`

Cree documentos de muestra

```
db.food.insertMany([
  { _id: 1, name: "Apple", special_diets: ["vegetarian", "gluten-free"] },
  { _id: 2, name: "Broccoli" },
  { _id: 3, name: "Chicken", special_diets: ["dairy-free"] }
]);
```

Cree un índice disperso en el campo `special_diets`

```
db.food.createIndex({ "special_diets": 1 }, { sparse: true, name:
 "special_diets_sparse_asc" });
```

Ejemplo de consulta

```
db.food.find({ "special_diets": { $exists: true } });
```

Salida

```
[
  { "_id" : 1, "name" : "Apple", "special_diets" : [ "vegetarian", "gluten-free" ] },
  { "_id" : 3, "name" : "Chicken", "special_diets" : [ "dairy-free" ] }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$exists` comando, elija la pestaña del idioma que desee usar:

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

Nuevo a partir de la versión 4.0.

Elastic Cluster no lo admite.

El `$expr` operador de Amazon DocumentDB permite utilizar expresiones de agregación en el lenguaje de consulta. Le permite realizar comparaciones y cálculos complejos en los campos de un documento, de forma similar a como utilizaría las etapas de canalización de agregación.

Los operadores`$eq`, `$lt` `$lte``$gt`, y de `$gte` comparación colocados en un `$expr` operador pueden usar un índice, incluido un índice de la `from` colección a la que se hace referencia en una `$lookup` etapa.

Limitaciones

- Los índices solo se pueden usar para comparar campos y constantes, por lo que el operando debe convertirse en una constante. Por ejemplo, una comparación entre un valor `$a` y una constante puede usar un índice, pero una comparación entre y `$a` no. `$b`
- No se utilizan índices multiclave, parciales o dispersos. En esos casos, la consulta vuelve a un escaneo de colección.
- Cuando el `let` operando da como resultado un valor vacío o faltante, la comparación no puede utilizar una búsqueda de índice específica. En este caso, Amazon DocumentDB realiza un escaneo completo del índice.

Parámetros

- `expression`: expresión que devuelve un valor booleano que permite realizar comparaciones y cálculos en los campos del documento.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$expr` operador para buscar todos los documentos en los que el `manufacturingCost` campo es mayor que el `price` campo.

Cree documentos de muestra

```
db.inventory.insertMany([
  { item: "abc", manufacturingCost: 500, price: 100 },
  { item: "def", manufacturingCost: 300, price: 450 },
  { item: "ghi", manufacturingCost: 400, price: 120 }
]);
```

Ejemplo de consulta

```
db.inventory.find({
  $expr: {
    $gt: ["$manufacturingCost", "$price"]
  }
})
```

Salida

```
{ "_id" : ObjectId("60b9d4d68d2cac581bc5a89a"), "item" : "abc", "manufacturingCost" :
 500, "price" : 100 },
{ "_id" : ObjectId("60b9d4d68d2cac581bc5a89c"), "item" : "ghi", "manufacturingCost" :
 400, "price" : 120 }
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$expr` comando, elige la pestaña del idioma que quieres usar:

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

El `$gt` operador de Amazon DocumentDB se utiliza para seleccionar documentos en los que el valor del campo especificado es mayor que el valor especificado. Este operador es útil para filtrar y consultar datos en función de comparaciones numéricas.

Parámetros

- `field`: El campo que se va a comparar.
- `value`: El valor con el que se va a comparar.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra cómo utilizar el `$gt` operador para buscar todos los documentos en los que el `age` campo es mayor que 30.

Cree documentos de muestra

```
db.users.insertMany([
  { name: "John", age: 25 },
  { name: "Jane", age: 32 },
  { name: "Bob", age: 45 },
  { name: "Alice", age: 28 }
]);
```

Ejemplo de consulta

```
db.users.find({ age: { $gt: 30 } });
```

Salida

```
{ "_id" : ObjectId("6249e5c22a5d39884a0a0001"), "name" : "Jane", "age" : 32 },
{ "_id" : ObjectId("6249e5c22a5d39884a0a0002"), "name" : "Bob", "age" : 45 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$gt` comando, elija la pestaña del idioma que desee usar:

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

El `$gte` operador de Amazon DocumentDB se utiliza para hacer coincidir valores superiores o iguales a un valor especificado. Este operador es útil para filtrar y consultar datos en función de comparaciones numéricas.

Parámetros

- `field`: el campo que se va a comparar con el valor proporcionado.
- `value`: el valor que se va a comparar con el campo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra el uso del `$gte` operador en Amazon DocumentDB para buscar todos los documentos en los que el campo «edad» es mayor o igual a 25.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.users.find({ age: { $gte: 25 } }, { _id: 0, name: 1, age: 1 });
```

Salida

```
{ "name" : "Jane", "age" : 25 }
{ "name" : "Bob", "age" : 30 }
{ "name" : "Alice", "age" : 35 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$gte` comando, elija la pestaña del idioma que desee usar:

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

El `$in` operador de Amazon DocumentDB es un operador de consulta lógico que permite buscar documentos en los que el valor de un campo es igual a cualquiera de los valores especificados en una matriz.

Parámetros

- `field`: el campo para compararlo con la matriz proporcionada.
- `[value1, value2, ...]`: una matriz de valores para compararlos con el campo especificado.

Dólar (**$**) en los nombres de campo

Consulte Dólar($) y punto(.) en los nombres de campos para conocer las limitaciones relacionadas con la consulta de campos con $ prefijo `$in` en objetos anidados.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$in` operador para buscar documentos en los que el `color` campo sea uno de los valores de la matriz proporcionada.

Cree documentos de muestra

```
db.colors.insertMany([
  { "_id": 1, "color": "red" },
  { "_id": 2, "color": "green" },
  { "_id": 3, "color": "blue" },
  { "_id": 4, "color": "yellow" },
  { "_id": 5, "color": "purple" }
])
```

Ejemplo de consulta

```
db.colors.find({ "color": { "$in": ["red", "blue", "purple"] } })
```

Salida

```
{ "_id": 1, "color": "red" },
{ "_id": 3, "color": "blue" },
{ "_id": 5, "color": "purple" }
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$in` comando, elige la pestaña del idioma que quieres usar:

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

Nuevo a partir de la versión 4.0.

No es compatible con el clúster de Elastic.

El `$jsonSchema` operador de Amazon DocumentDB se utiliza para filtrar documentos en función de un esquema JSON específico. Este operador le permite consultar documentos que coinciden con un esquema JSON concreto, lo que garantiza que los documentos recuperados cumplan con requisitos estructurales y de tipo de datos específicos.

Al utilizar el operador de consulta de `$jsonSchema` evaluación como parte de la creación de una colección, puede validar el esquema de los documentos que se van a insertar en la colección. Para obtener información adicional, consulte Cómo utilizar la validación de esquemas JSON.

Parámetros

- `required`(matriz): especifica los campos obligatorios del documento.
- `properties`(objeto): define el tipo de datos y otras restricciones para cada campo del documento.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo se muestra el uso del `$jsonSchema` operador para filtrar la `employees` colección y recuperar únicamente los documentos que tengan los `age` campos `employeeId` y`name`, si el `employeeId` campo es del tipo. `string`

Cree documentos de muestra

```
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" }
]);
```

Ejemplo de consulta

```
db.employees.aggregate([
  { $match: {
    $jsonSchema: {
      required: ["name", "employeeId", "age"],
      properties: { "employeeId": { "bsonType": "string" } }
    }
  }}
]);
```

Salida

```
{ "_id" : ObjectId("6908e8b61f77fc26b2ecd26f"), "name" : { "firstName" : "Emily",
 "lastName" : "Brown" }, "employeeId" : "2", "age" : 25 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$jsonSchema` comando, elija la pestaña del idioma que desee usar:

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

El `$lt` operador de Amazon DocumentDB se utiliza para hacer coincidir valores inferiores al valor especificado. Este operador es útil para filtrar y consultar datos en función de comparaciones numéricas.

Parámetros

- `field`: el nombre del campo que se va a comprobar.
- `value`: el valor con el que se va a comparar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$lt` operador para recuperar documentos cuyo `Inventory.OnHand` valor es inferior a 50.

Cree documentos de muestra

```
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"] }
]);
```

Ejemplo de consulta

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

Salida

```
{ "_id" : 1, "Inventory" : { "OnHand" : 25, "Sold" : 60 }, "Colors" : [ "Red",
 "Blue" ] }
{ "_id" : 3, "Inventory" : { "OnHand" : 10, "Sold" : 85 }, "Colors" : [ "Red",
 "Green" ] }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$lt` comando, elija la pestaña del idioma que desee usar:

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

El operador de $mod consulta selecciona los documentos en los que un valor de campo dividido por un divisor tiene un resto especificado. Esto resulta útil para filtrar documentos en función de las condiciones aritméticas de los módulos.

Parámetros

- `divisor`: el número por el que se va a dividir.
- `remainder`: El valor restante esperado.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra el uso del $mod operador para buscar todos los pedidos en los que la cantidad es un número impar.

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.orders.find({ quantity: { $mod: [2, 1] } });
```

Salida

```
{ "_id" : 1, "item" : "Widget", "quantity" : 15 }
```

```
{ "_id" : 3, "item" : "Tool", "quantity" : 7 }
{ "_id" : 5, "item" : "Part", "quantity" : 9 }
```

Esta consulta devuelve documentos en los que la cantidad dividida por 2 tiene un resto igual a 1, lo que permite seleccionar todas las cantidades impares.

Ejemplos de código

Para ver un ejemplo de código para usar el operador de $mod consulta, elija la pestaña correspondiente al idioma que desee usar:

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

El `$lte` operador de Amazon DocumentDB se utiliza para hacer coincidir documentos en los que el valor de un campo especificado es inferior o igual al valor especificado. Este operador es útil para filtrar y consultar datos en función de comparaciones numéricas.

Parámetros

- `field`: El campo que se va a comparar.
- `value`: El valor con el que se va a comparar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$lte` operador para recuperar documentos en los que el `quantity` campo es menor o igual a 10.

Cree documentos de muestra

```
db.inventory.insertMany([
  { item: "canvas", qty: 100 },
  { item: "paint", qty: 50 },
  { item: "brush", qty: 10 },
  { item: "paper", qty: 5 }
]);
```

Ejemplo de consulta

```
db.inventory.find({ qty: { $lte: 10 } });
```

Salida

```
{ "_id" : ObjectId("..."), "item" : "brush", "qty" : 10 },
```

```
{ "_id" : ObjectId("..."), "item" : "paper", "qty" : 5 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$lte` comando, elija la pestaña del idioma que desee usar:

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

El `$meta` operador se utiliza para acceder a los metadatos asociados a la ejecución de la consulta actual. Este operador se utiliza principalmente para las operaciones de búsqueda de texto, en las que los metadatos pueden proporcionar información sobre la relevancia de los documentos coincidentes.

Parámetros

- `textScore`: recupera la puntuación de búsqueda de texto del documento. Esta puntuación indica la relevancia del documento para la consulta de búsqueda de texto.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$meta` operador para recuperar la puntuación de búsqueda de texto de los documentos que coinciden con una consulta de búsqueda de texto.

Cree documentos de muestra

```
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 índice de texto

```
db.documents.createIndex({ content: "text" });
```

Ejemplo de consulta

```
db.documents.find(
  { $text: { $search: "coffee" } },
  { _id: 0, title: 1, content: 1, score: { $meta: "textScore" } }
).sort({ score: { $meta: "textScore" } });
```

Salida

```
[
  {
    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
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el $meta comando, elija la pestaña del idioma que desee usar:

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

El `$ne` operador se usa para hacer coincidir documentos en los que el valor de un campo no es igual al valor especificado. Es un operador de comparación que se puede usar en los predicados de consulta para filtrar documentos.

La versión 2.0 de Planner agregó compatibilidad con índices para`$ne`.

Parámetros

- `field`: El campo que se va a comprobar.
- `value`: El valor con el que se va a comparar.

Ejemplo (MongoDB Shell)

En este ejemplo, encontraremos todos los documentos de la `users` colección en los que el `status` campo no sea igual a. `"active"`

Crea documentos de muestra

```
db.users.insertMany([
  { name: "John", status: "active" },
  { name: "Jane", status: "inactive" },
  { name: "Bob", status: "suspended" },
  { name: "Alice", status: "active" }
]);
```

Ejemplo de consulta

```
db.users.find({ status: { $ne: "active" } });
```

Salida

```
[
  {
    _id: ObjectId('...'),
    name: 'Jane',
    status: 'inactive'
  },
  {
    _id: ObjectId('...'),
    name: 'Bob',
    status: 'suspended'
  }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$ne` comando, elige la pestaña del idioma que quieres usar:

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

El `$nin` operador se usa para hacer coincidir valores que no están en la matriz especificada. Es el inverso del `$in` operador, que coincide con los valores que están en la matriz especificada.

La versión 2.0 de Planner agregó compatibilidad con índices para`$nin`.

Parámetros

- `field`: El campo que se va a comprobar.
- `array`: El conjunto de valores con los que se va a realizar la comprobación.

Dollar (**$**) en los nombres de los campos

Consulte las limitaciones Dólar($) y punto(.) en los nombres de campos relacionadas con la consulta de campos con $ prefijo `$nin` en objetos anidados.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$nin` operador para buscar documentos en los que el `category` campo no sea igual a «Ficción» o «Misterio».

Cree documentos de muestra

```
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" }
])
```

Ejemplo de consulta

```
db.books.find({
  category: {
    $nin: ["Fiction", "Mystery"]
  }
})
```

Salida

```
[
  {
    _id: ObjectId('...'),
    title: 'The Martian',
    author: 'Andy Weir',
    category: 'Science Fiction'
  },
  {
    _id: ObjectId('...'),
    title: 'The Alchemist',
    author: 'Paulo Coelho',
    category: 'Philosophy'
  }
```

```
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$nin` comando, elija la pestaña del idioma que desee usar:

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

El `$nor` operador se utiliza para hacer coincidir documentos en los que ninguna de las condiciones de consulta especificadas es verdadera. Es similar a la operación lógica «NOR», en la que el resultado es verdadero si ninguno de los operandos es verdadero.

Parámetros

- `expression1`: la primera expresión que se va a evaluar.
- `expression2`: La segunda expresión que se va a evaluar.
- `expressionN`: Expresiones adicionales para evaluar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$nor` operador al recuperar documentos en los que el `qty` campo no es inferior a 20 ni igual a «XL». `size`

Cree documentos de muestra

```
db.items.insertMany([
  { qty: 10, size: "M" },
  { qty: 15, size: "XL" },
  { qty: 25, size: "L" },
  { qty: 30, size: "XL" }
])
```

Ejemplo de consulta

```
db.items.find({
  $nor: [
    { qty: { $lt: 20 } },
    { size: "XL" }
  ]
})
```

Salida

```
[
  { "_id" : ObjectId("..."), "qty" : 25, "size" : "L" }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$nor` comando, elija la pestaña del idioma que desee usar:

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

El `$not` operador se usa para negar el resultado de una expresión dada. Le permite seleccionar documentos en los que la condición especificada no coincide.

La versión 2.0 de Planner agregó compatibilidad con índices para `$not {eq}` y`$not {in}`.

Parámetros

- `expression`: La expresión para negar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$not` operador para buscar documentos en los que el `status` campo no es igual a «activo».

Cree documentos de muestra

```
db.users.insertMany([
  { name: "John", status: "active" },
  { name: "Jane", status: "inactive" },
  { name: "Bob", status: "pending" },
  { name: "Alice", status: "active" }
]);
```

Ejemplo de consulta

```
db.users.find({ status: { $not: { $eq: "active" } } });
```

Salida

```
[
  {
    _id: ObjectId('...'),
    name: 'Jane',
    status: 'inactive'
  },
  {
    _id: ObjectId('...'),
    name: 'Bob',
    status: 'pending'
  }
]
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$not` comando, elija la pestaña del idioma que desee usar:

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

El `$or` operador se usa para realizar una operación OR lógica en una matriz de dos o más expresiones. Devuelve documentos que coinciden con al menos una de las expresiones. Este operador resulta útil cuando necesita consultar documentos que cumplan alguna de las múltiples condiciones.

Parámetros

- `expression1`: la primera expresión que se va a evaluar.
- `expression2`: La segunda expresión que se va a evaluar.
- `...`: Expresiones adicionales para evaluar (opcional).

Ejemplo (MongoDB Shell)

En el siguiente ejemplo se muestra el uso del `$or` operador para buscar documentos en los que el modelo «Heavy H1" `make` esSportForYou"» con el modelo «Bolid 1". TruckForYou

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.cars.find({
  $or: [
    { make: "TruckForYou", model: "Heavy H1" },
    { make: "SportForYou", model: "Bolid 1" }
  ]
});
```

Salida

```
[
  {
    _id: ObjectId('...'),
    make: 'TruckForYou',
    model: 'Heavy H1',
    year: 2020
  },
  {
    _id: ObjectId('...'),
    make: 'SportForYou',
    model: 'Bolid 1',
    year: 2021
  }
]
```

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$or` comando, elige la pestaña del idioma que quieres usar:

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

El `$regex` operador le permite realizar coincidencias de expresiones regulares en campos de cadenas. Es una potente herramienta para buscar y filtrar documentos en función de patrones complejos.

Parámetros

- `regular expression`: El patrón de expresión regular que se debe comparar con el campo.
- `$options`: (opcional) Ofrece opciones para modificar el comportamiento de búsqueda, como distinguir entre mayúsculas y minúsculas, la coincidencia global, etc.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$regex` operador para buscar documentos en los que el campo «nombre» coincide con un patrón específico.

Cree documentos de muestra

```
db.users.insertMany([
  { name: "John Doe" },
  { name: "Jane Smith" },
  { name: "Alice Johnson" },
  { name: "Bob Williams" },
  { name: "Charlie Davis" }
]);
```

Ejemplo de consulta

```
db.users.find({ name: { $regex: /^A/ } })
```

Salida

```
[
```

```
  { "_id" : ObjectId("..."), "name" : "Alice Johnson" }
]
```

Esta consulta devolverá todos los documentos en los que el campo «nombre» comience por la letra «A».

Ejemplos de código

Para ver un ejemplo de código para usar el `$regex` comando, elija la pestaña correspondiente al idioma que desee usar:

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()
```

$sampleRate

Nuevo desde la versión 8.0.1.

Elastic Cluster no lo admite.

El `$sampleRate` operador de Amazon DocumentDB compara una muestra aleatoria de los documentos de entrada en función de la tasa especificada. Puede usarlo en un filtro de `find()` consultas o en una `$match` etapa de agregación. Como la selección es probabilística, la cantidad de documentos devueltos es aproximada y puede variar de una tirada a otra.

Parámetros

- `value`: un número del rango `[0, 1]` (incluido), que se pasa directamente al operador (por ejemplo,`{ $sampleRate: 0.3 }`), que establece la probabilidad de incluir cada documento. El valor de no `0` devuelve ningún documento; el valor de 1 devuelve todos los documentos.

Ejemplo (MongoDB Shell)

El siguiente ejemplo se utiliza `$sampleRate` para devolver aproximadamente el 30 por ciento de los documentos de una colección.

Cree documentos de muestra

```
db.events.insertMany([
  { _id: 1, type: "click" },
  { _id: 2, type: "view" },
  { _id: 3, type: "click" },
  { _id: 4, type: "view" },
  { _id: 5, type: "purchase" }
]);
```

Ejemplo de consulta

```
db.events.aggregate([
  { $match: { $sampleRate: 0.3 } }
]);
```

Salida

La operación devuelve un subconjunto aleatorio de los documentos. Como `$sampleRate` es probabilístico, los documentos específicos devueltos y su recuento varían cada vez que se ejecuta la consulta.

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$sampleRate` comando, elija la pestaña correspondiente al idioma que desee utilizar:

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

El operador `$slice` de proyección limita el número de elementos de matriz devueltos en el resultado de una consulta. Permite recuperar un número específico de elementos desde el principio o el final de un campo de matriz sin cargar toda la matriz.

Parámetros

- `field`: El campo de matriz que se va a proyectar.
- `count`: Número de elementos que se van a devolver. Los valores positivos devuelven los elementos desde el principio y los negativos desde el final.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el operador de `$slice` proyección para devolver solo los dos primeros elementos de un campo matricial.

Cree documentos de muestra

```
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"] }
]);
```

Ejemplo de consulta

```
db.inventory.find(
  {},
  { item: 1, tags: { $slice: 2 } }
)
```

Salida

```
{ "_id" : 1, "item" : "notebook", "tags" : [ "office", "school" ] }
{ "_id" : 2, "item" : "pen", "tags" : [ "office", "writing" ] }
{ "_id" : 3, "item" : "folder", "tags" : [ "office", "supplies" ] }
```

Ejemplos de código

Para ver un ejemplo de código para usar el operador de `$slice` proyección, elija la pestaña del idioma que desee usar:

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

El operador de `$size` consulta hace coincidir los documentos en los que un campo de matriz tiene exactamente el número de elementos especificado. Esto resulta útil para filtrar documentos en función de la longitud de la matriz.

Parámetros

- `field`: El campo de matriz que se va a comprobar.
- `count`: El número exacto de elementos que debe contener la matriz.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra el uso `$size` del operador para buscar todos los productos que tienen exactamente tres etiquetas.

Cree documentos de muestra

```
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"] }
]);
```

Ejemplo de consulta

```
db.products.find({ tags: { $size: 3 } });
```

Salida

```
{ "_id" : 1, "name" : "Laptop", "tags" : [ "electronics", "computers", "portable" ] }
{ "_id" : 3, "name" : "Desk", "tags" : [ "furniture", "office", "workspace" ] }
```

Ejemplos de código

Para ver un ejemplo de código para usar el operador de `$size` consulta, elija la pestaña del idioma que desee usar:

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

El `$text` operador se usa para realizar búsquedas de texto completo en campos indexados por texto de una colección de documentos. Este operador permite buscar documentos que contienen palabras o frases específicas y se puede combinar con otros operadores de consulta para filtrar los resultados en función de criterios adicionales.

Parámetros

- `$search`: la cadena de texto que se va a buscar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo usar el `$text` operador para buscar documentos que contengan la palabra «interés» y filtrar los resultados en función del campo «star_rating».

Cree documentos de muestra

```
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" }
]);
```

Crear índice de texto

```
db.test.createIndex({ comments: "text" });
```

Ejemplo de consulta

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "interest"}}]})
```

Salida

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
{ "_id" : 6, "star_rating" : 5, "comments" : "interested in couch for sale, year
 2022" }
```

El comando anterior devuelve documentos con un campo indexado por texto que contiene cualquier forma de «interés» y una «star_rating» igual a 5.

Ejemplos de código

Para ver un ejemplo de código sobre el uso del `$text` comando, elige la pestaña correspondiente al idioma que quieres usar:

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

El `$type` operador se utiliza para comprobar el tipo de datos de un campo de un documento. Se puede utilizar cuando se necesitan operaciones o validaciones de un tipo específico. El `$type`

operador devuelve el tipo BSON de la expresión evaluada. El tipo devuelto es una cadena, que corresponde al tipo de campo o expresión.

La versión 2.0 de Planner agregó compatibilidad con índices para`$type`.

Parámetros

- `expression`: La expresión que se va a evaluar.

Ejemplo (MongoDB Shell)

Cree documentos de muestra

```
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 }
]);
```

Ejemplo de consulta

```
db.documents.find({
  $or: [
    { age: { $type: "number" } },
    { email: { $type: "string" } },
    { name: { $type: "string" } }
  ]
})
```

Salida

```
[
  { "_id": 1, "name": "John", "age": 30, "email": "john@example.com" },
  { "_id": 2, "name": "Jane", "age": "25", "email": 123456 }
]
```

Esta consulta devolverá los documentos en los que el `age` campo es del tipo «número», el `email` campo es del tipo «cadena» y el `name` campo es del tipo «cadena».

Ejemplos de código

Para ver un ejemplo de código para usar el `$type` comando, elija la pestaña correspondiente al idioma que desee usar:

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()
```

Operadores de actualización

En esta sección se proporciona información detallada sobre los operadores de actualización compatibles con Amazon DocumentDB.

Temas

- $
- $[]
- $ [<identificador>]
- $añadir ToSet
- $bit
- $currentDate
- $each
- $inc
- $max
- $min
- $mul
- $pop
- $position
- $pull
- $pullAll
- $push
- $rename

- $set
- conjunto $ OnInsert
- $slice
- $sort
- $unset

$

El operador $ posicional actualiza el primer elemento de la matriz que coincide con la condición de consulta. Actúa como marcador de posición para la posición del elemento de matriz coincidente.

Parámetros

- `field.$`: El campo de matriz con el operador posicional para actualizar el primer elemento coincidente.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del operador $ posicional para actualizar un elemento de matriz específico.

Cree documentos de muestra

```
db.inventory.insertMany([
  { _id: 1, item: "Widget", quantities: [10, 20, 30] },
  { _id: 2, item: "Gadget", quantities: [5, 15, 25] }
]);
```

Ejemplo de consulta

```
db.inventory.updateOne(
  { _id: 1, quantities: 20 },
  { $set: { "quantities.$": 22 } }
);
```

Salida

```
{
```

```
  "_id" : 1,
  "item" : "Widget",
  "quantities" : [ 10, 22, 30 ]
}
```

Ejemplos de código

Para ver un ejemplo de código para usar el operador $ posicional, elija la pestaña correspondiente al idioma que desee usar:

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()
```

$[]

El operador `$[]` all positional actualiza todos los elementos de una matriz. Se utiliza cuando se necesita modificar todos los elementos de un campo de matriz.

Parámetros

- `field.$[]`: El campo de matriz con el operador totalmente posicional para actualizar todos los elementos.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$[]` operador para actualizar todos los elementos de la matriz.

Cree documentos de muestra

```
db.products.insertOne({
  _id: 1,
  name: "Laptop",
  prices: [1000, 1100, 1200]
});
```

Ejemplo de consulta

```
db.products.updateOne(
  { _id: 1 },
```

```
  { $inc: { "prices.$[]": 50 } }
);
```

Salida

```
{
  "_id" : 1,
  "name" : "Laptop",
  "prices" : [ 1050, 1150, 1250 ]
}
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$[]` operador, elija la pestaña del idioma que desee usar:

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()
```

$ [<identificador>]

El operador posicional `$[<identifier>]` filtrado actualiza todos los elementos de la matriz que coinciden con las condiciones de filtro especificadas. Se utiliza con la `arrayFilters` opción de actualizar selectivamente los elementos de la matriz.

Parámetros

- `field.$[identifier]`: El campo de matriz con el operador posicional filtrado.
- `arrayFilters`: un conjunto de condiciones de filtro que determinan qué elementos se van a actualizar.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$[<identifier>]` operador para actualizar elementos de matriz específicos en función de una condición.

Cree documentos de muestra

```
db.students.insertOne({
```

```
  _id: 1,
  name: "Alice",
  grades: [
    { subject: "Math", score: 85 },
    { subject: "Science", score: 92 },
    { subject: "History", score: 78 }
  ]
});
```

Ejemplo de consulta

```
db.students.updateOne(
  { _id: 1 },
  { $inc: { "grades.$[elem].score": 5 } },
  { arrayFilters: [{ "elem.score": { $gte: 80 } }] }
);
```

Salida

```
{
  "_id" : 1,
  "name" : "Alice",
  "grades" : [
    { "subject" : "Math", "score" : 90 },
    { "subject" : "Science", "score" : 97 },
    { "subject" : "History", "score" : 78 }
  ]
}
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$[<identifier>]` operador, elija la pestaña del idioma que desee usar:

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()
```

$añadir ToSet

El `$addToSet` operador de Amazon DocumentDB se utiliza para añadir un valor a una matriz solo si el valor aún no está presente en la matriz. Esto resulta útil para garantizar que una matriz contenga elementos únicos.

Parámetros

- `field`: el campo que se va a actualizar.
- `value`: el valor que se va a añadir al campo de matriz. Puede ser un valor único o una expresión.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$addToSet` operador para añadir elementos únicos a una matriz.

Cree documentos de muestra

```
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"] }
])
```

Ejemplo de consulta

```
db.products.update(
  { "item": "apple" },
  { $addToSet: { "tags": "green" } }
)
```

Salida

```
{ "_id": 1, "item": "apple", "tags": ["fruit", "red", "round", "green"] }
```

En este ejemplo, el `$addToSet` operador añade la etiqueta «verde» a la matriz de «etiquetas» del documento, donde el campo «elemento» es «manzana». Como la palabra «verde» aún no estaba en la matriz, se agregó.

Ejemplos de código

Para ver un ejemplo de código para usar el `$addToSet` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$bit` operador de Amazon DocumentDB le permite realizar operaciones bit a bit en los bits de un campo determinado. Esto puede resultar útil para tareas como configurar, borrar o comprobar el estado de los bits individuales de un número.

Parámetros

- `field`: El campo en el que se realizan operaciones bit a bit.
- `and`: un valor entero que se utiliza para realizar una operación AND bit a bit en el campo.
- `or`: un valor entero que se utiliza para realizar una operación OR bit a bit en el campo.
- `xor`: un valor entero que se utiliza para realizar una operación XOR bit a bit en el campo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$bit` operador para realizar operaciones bit a bit en un campo numérico.

Cree documentos de muestra

```
db.numbers.insert([
  { "_id": 1, "number": 5 },
  { "_id": 2, "number": 12 }
])
```

Ejemplo de consulta

```
db.numbers.update(
  { "_id": 1 },
```

```
  { "$bit": { "number": { "and": 3 } } }
)
```

Salida

```
{
  "_id": 1,
  "number": 1
}
```

En este ejemplo, el `$bit` operador se utiliza para realizar una operación AND bit a bit en el campo «numérico» del documento con un valor `_id` de 1. El resultado es que el valor del campo «numérico» se establece en 1, que es el resultado de la operación AND bit a bit entre el valor original de 5 y el valor 3.

Ejemplos de código

Para ver un ejemplo de código para usar el `$bit` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$currentDate` operador se utiliza para establecer el valor de un campo en la fecha y hora actuales. Este operador es útil para actualizar automáticamente un campo con la marca de tiempo actual cuando se inserta o actualiza un documento.

Parámetros

- `field`: el campo que se va a actualizar con la fecha y hora actuales.
- `type`: (opcional) Especifica el tipo de BSON que se utilizará en la fecha actual. Puede ser `date` o `timestamp`.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra cómo utilizar el `$currentDate` operador para establecer el `lastModified` campo en la fecha y hora actuales al insertar un documento nuevo.

Cree documentos de muestra

```
db.users.insert({
  name: "John Doe",
  email: "john.doe@example.com"
})
```

Ejemplo de consulta

```
db.users.updateOne(
  { name: "John Doe" },
  { $currentDate: { lastModified: true } }
)
```

Ver documento actualizado

```
db.users.findOne({ name: "John Doe" })
```

Salida

```
{
  _id: ObjectId('...'),
  name: 'John Doe',
  email: 'john.doe@example.com',
  lastModified: ISODate('2025-10-25T22:50:29.963Z')
}
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$currentDate` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$each` operador se utiliza junto con otros operadores de actualización, como `$push` y`$addToSet`, para añadir varios valores a un campo de matriz. Permite añadir varios elementos a una matriz en una sola operación, en lugar de tener que ejecutar varias operaciones de actualización.

Parámetros

- `value`: La matriz de valores que se van a añadir al campo de la matriz.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra el uso `$each` del operador con el `$push` operador para añadir varios elementos a un campo de matriz.

Cree documentos de muestra

```
db.fruits.insertOne({
  _id: 1,
  fruits: ["apple", "banana"]
})
```

Ejemplo de consulta

```
db.fruits.updateOne(
  { _id: 1 },
  { $push: { fruits: { $each: ["cherry", "durian", "elderberry"] } } }
)
```

Ver documento actualizado

```
db.fruits.findOne({ _id: 1 })
```

Salida

```
{
  _id: 1,
  fruits: [ 'apple', 'banana', 'cherry', 'durian', 'elderberry' ]
}
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$each` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$inc` operador se usa para incrementar el valor de un campo en una cantidad específica. Se utiliza para actualizar un campo numérico, como un contador o una calificación, sin tener que recuperar el valor actual, calcular el nuevo valor y, a continuación, actualizar el campo.

Parámetros

- `field`: el nombre del campo que se va a incrementar.
- `amount`: la cantidad en la que se va a incrementar el campo. Puede ser un valor positivo o negativo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$inc` operador para incrementar el `age` campo de un documento.

Cree documentos de muestra

```
db.users.insertOne({_id: 123, name: "John Doe", age: 30})
```

Ejemplo de consulta

```
db.users.updateOne({_id: 123}, {$inc: {age: 1}})
```

Ver documento actualizado

```
db.users.findOne({_id: 123})
```

Salida

```
{ "_id" : 123, "name" : "John Doe", "age" : 31 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$inc` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El operador de `$max` actualización actualiza el valor de un campo solo si el valor especificado es mayor que el valor del campo actual. Este operador es útil para mantener los valores máximos en todas las actualizaciones.

Parámetros

- `field`: el campo que se va a actualizar.
- `value`: el valor que se va a comparar con el valor del campo actual.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra el uso `$max` del operador para actualizar la puntuación más alta registrada para un jugador.

Cree documentos de muestra

```
db.scores.insertMany([
  { _id: 1, player: "Alice", highScore: 85 },
  { _id: 2, player: "Bob", highScore: 92 },
  { _id: 3, player: "Charlie", highScore: 78 }
])
```

Ejemplo de actualización

```
db.scores.updateOne(
  { _id: 1 },
  { $max: { highScore: 95 } }
)
```

Resultado

El `highScore` campo de Alice se actualiza a 95 porque 95 es mayor que el valor actual de 85.

```
{ "_id": 1, "player": "Alice", "highScore": 95 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$max` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El operador de `$min` actualización actualiza el valor de un campo solo si el valor especificado es inferior al valor del campo actual. Este operador es útil para mantener los valores mínimos en todas las actualizaciones.

Parámetros

- `field`: el campo que se va a actualizar.
- `value`: el valor que se va a comparar con el valor del campo actual.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$min` operador para actualizar la temperatura más baja registrada en una estación meteorológica.

Cree documentos de muestra

```
db.weather.insertMany([
  { _id: 1, station: "Station A", lowestTemp: 15 },
  { _id: 2, station: "Station B", lowestTemp: 20 },
  { _id: 3, station: "Station C", lowestTemp: 18 }
])
```

Ejemplo de actualización

```
db.weather.updateOne(
  { _id: 1 },
  { $min: { lowestTemp: 12 } }
)
```

Resultado

El `lowestTemp` campo de la estación A se actualiza a 12 porque 12 es menor que el valor actual de 15.

```
{ "_id": 1, "station": "Station A", "lowestTemp": 12 }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$min` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$mul` operador de Amazon DocumentDB se utiliza para multiplicar el valor de un campo por un número específico. Esto puede resultar útil para actualizar varios documentos de forma atómica y coherente, por ejemplo, para actualizar las millas de vuelo en función del estado de una tarjeta de crédito.

Parámetros

- `field`: El campo que se va a multiplicar.
- `multiplier`: el número por el que se va a multiplicar el valor del campo.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo utilizar el `$mul` operador para duplicar el `flight_miles` valor de todos los documentos en los que se encuentra el `credit_card` campo. `true`

Cree documentos de muestra

```
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] }
]);
```

Ejemplo de consulta

```
db.miles.update(
  { "credit_card": { "$eq": true } },
  { "$mul": { "flight_miles.$[]": NumberInt(2) } },
  { "multi": true }
);
```

Salida

```
{ "_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 ] }
```

Para los clientes que tienen una tarjeta de crédito, sus millas de vuelo se han duplicado.

El operador de matriz `$[]` posicional se utiliza para aplicar la `$mul` operación a cada elemento de la `flight_miles` matriz.

Ejemplos de código

Para ver un ejemplo de código para usar el `$mul` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El $pop operador de Amazon DocumentDB se utiliza para eliminar el primer o el último elemento de un campo de matriz. Resulta especialmente útil cuando necesita mantener una matriz de tamaño fijo o implementar una estructura de datos similar a una cola en un documento.

Parámetros

- `field`: el nombre del campo de matriz del que se va a eliminar un elemento.
- `value`: un valor entero que determina la posición del elemento que se va a eliminar. Un valor de `1` elimina el último elemento, mientras que un valor de `-1` elimina el primer elemento.

Ejemplo (MongoDB Shell)

En este ejemplo se muestra cómo utilizar el $pop operador para eliminar el primer y el último elemento de un campo matricial.

Cree documentos de muestra

```
db.users.insertMany([
  { "_id": 1, "name": "John Doe", "hobbies": ["reading", "swimming", "hiking"] },
  { "_id": 2, "name": "Jane Smith", "hobbies": ["cooking", "gardening", "painting"] }
])
```

Ejemplo de consulta

```
// 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 } })
```

Salida

```
{ "_id" : 1, "name" : "John Doe", "hobbies" : [ "swimming", "hiking" ] }
{ "_id" : 2, "name" : "Jane Smith", "hobbies" : [ "cooking", "gardening" ] }
```

Ejemplos de código

Para ver un ejemplo de código para usar el $pop comando, elija la pestaña del idioma que desee usar:

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

El `$position` modificador de Amazon DocumentDB especifica la ubicación de la matriz en la que el operador inserta `$push` los elementos. Sin el `$position` modificador, el `$push` operador inserta elementos al final de la matriz.

Parámetros

- `field`: El campo de matriz que se va a actualizar.
- num: la posición de la matriz en la que se deben insertar los elementos, según la indexación basada en cero.

Nota: Para usar el `$position` modificador, debe aparecer junto con el modificador. `$each`

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra cómo utilizar el `$position` operador para insertar tareas en posiciones específicas de un sistema de gestión de proyectos.

Cree documentos de muestra

```
db.projects.insertOne({ "_id": 1, "name": "Website Redesign", "tasks": ["Design
 mockups"] })
```

Ejemplo de consulta 1: añadir tareas urgentes al principio

```
db.projects.updateOne(
   { _id: 1 },
   {
     $push: {
        tasks: {
           $each: ["Security audit", "Performance review"],
           $position: 0
        }
     }
   }
)
```

Resultado 1

```
{ "_id": 1, "name": "Website Redesign", "tasks": ["Security audit", "Performance
 review", "Design mockups"] }
```

Ejemplo de consulta 2: añadir tareas en una posición específica

```
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
         }
      }
   }
)
```

Resultado 2

```
{ "_id": 2, "name": "Mobile App", "tasks": ["Setup project", "Create wireframes", "Code
 review", "Testing phase", "Deploy to store"] }
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$position` comando, elija la pestaña del idioma que desee usar:

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

El `$pull` operador se utiliza para eliminar de una matriz todas las instancias de un valor o valores que coincidan con una condición especificada. Este operador resulta útil cuando se necesitan eliminar elementos específicos de un campo matricial de un documento.

Parámetros

- `field`: el nombre del campo de matriz del que se van a eliminar los valores.
- `value`: el valor o la condición que determina los elementos que se van a eliminar de la matriz.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra cómo utilizar el `$pull` operador para eliminar elementos de un campo matricial.

Cree documentos de muestra

```
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"]
  }
])
```

Ejemplo de consulta

```
db.restaurants.updateMany(
  { cuisine: "Italian" },
  { $pull: { features: "Takeout" } }
)
```

Salida

```
{
  "acknowledged" : true,
  "matchedCount" : 1,
  "modifiedCount" : 1
```

```
}
```

La consulta anterior elimina la función «Comida para llevar» de todos los documentos en los que el `cuisine` campo es «italiano».

Ejemplos de código

Para ver un ejemplo de código para usar el `$pull` comando, selecciona la pestaña correspondiente al idioma que deseas usar:

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

El `$pullAll` operador de Amazon DocumentDB se utiliza para eliminar todas las instancias de los valores especificados de un campo de matriz. Esto resulta especialmente útil cuando necesita eliminar varios elementos de una matriz en una sola operación.

Parámetros

- `field`: el nombre del campo de matriz del que se van a eliminar los elementos.
- `value`: una matriz de valores para eliminar del campo de matriz.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$pullAll` operador para eliminar varios elementos de un campo matricial.

Cree documentos de muestra

```
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"]
  }
])
```

Ejemplo de consulta

```
db.restaurants.update(
  { "name": "Taj Mahal" },
  { $pullAll: { "features": ["Private Dining", "Live Music"] } }
)
```

Salida

```
{
  "name": "Taj Mahal",
  "cuisine": "Indian",
  "features": []
}
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$pullAll` comando, elija la pestaña del idioma que desee usar:

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

El `$push` operador de Amazon DocumentDB se utiliza para añadir un elemento a un campo de matriz de un documento. Este operador resulta especialmente útil cuando necesita añadir nuevos datos a una matriz existente sin sobrescribir toda la matriz.

Parámetros

- `field`: el nombre del campo de matriz al que se debe añadir el nuevo elemento.
- `value`: El valor que se va a añadir a la matriz.
- `position`: (opcional) Un modificador que especifica la posición de la matriz en la que se debe añadir el nuevo elemento. Los modificadores admitidos incluyen $ (añadir al final de la matriz) y `$[]` (añadir al final de la matriz, ignorando cualquier filtro de matriz).

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$push` operador para añadir nuevos elementos a un campo matricial de un documento.

Cree documentos de muestra

```
db.users.insert([
  { _id: 1, name: "John Doe", hobbies: ["reading", "swimming"] },
  { _id: 2, name: "Jane Smith", hobbies: ["gardening", "cooking"] }
])
```

Ejemplo de consulta

```
db.users.updateOne(
  { _id: 1 },
  { $push: { hobbies: "hiking" } }
)
```

Salida

```
{
  "acknowledged" : true,
  "matchedCount" : 1,
  "modifiedCount" : 1
}
```

Tras ejecutar la actualización, el documento con la `hobbies` matriz se `_id: 1` actualizará `a["reading", "swimming", "hiking"]`.

Ejemplos de código

Para ver un ejemplo de código para usar el `$push` comando, elija la pestaña correspondiente al idioma que desee usar:

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

El `$rename` operador de Amazon DocumentDB se utiliza para cambiar el nombre de un campo de un documento. Este operador puede resultar especialmente útil cuando necesita actualizar la estructura de sus documentos o alinearlos con nuevos modelos de datos.

Parámetros

- `field`: el campo al que se va a cambiar el nombre.
- `newName`: el nuevo nombre del campo.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo se muestra cómo utilizar el `$rename` operador para cambiar el nombre del `"Date.DoW"` campo a `"Date.DayOfWeek"` en un documento con el `"DocName"` campo establecido en. `"Document 1"`

Cree documentos de muestra

```
db.example.insertOne({
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    },
    "Words": 2482
})
```

Ejemplo de consulta

```
db.example.update(
    { "DocName": "Document 1" },
    { $rename: { "Date.DoW": "Date.DayOfWeek" } }
)
```

Salida

```
{
```

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

Ejemplos de código

Para ver un ejemplo de código para usar el `$rename` comando, elija la pestaña del idioma que desee usar:

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

El `$set` operador de Amazon DocumentDB se utiliza para actualizar el valor de un campo específico de un documento. Este operador le permite añadir nuevos campos o modificar los existentes en un documento. Es un operador de actualización fundamental en el controlador Java de MongoDB, que es compatible con Amazon DocumentDB.

Parámetros

- `field`: El campo que se va a actualizar.
- `value`: el nuevo valor del campo.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra cómo utilizar el `$set` operador para actualizar el `Item` campo de un documento.

Cree documentos de muestra

```
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
    }
  }
])
```

Ejemplo de consulta

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

Salida

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

Ejemplos de código

Para ver un ejemplo de código para usar el `$set` comando, elija la pestaña del idioma que desee usar:

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()
```

conjunto $ OnInsert

El `$setOnInsert` operador de Amazon DocumentDB se usa para establecer el valor de un campo si se inserta un documento, pero no tiene ningún efecto si el documento se actualiza.

Parámetros

- `field`: El campo que se va a configurar.
- `value`: el valor que se va a asignar al campo.

Ejemplo (MongoDB Shell)

El siguiente ejemplo demuestra el uso del `$setOnInsert` operador en Amazon DocumentDB. Crea un documento nuevo si el documento aún no existe, pero no tiene ningún efecto si el documento se está actualizando.

Crea documentos de muestra

```
db.users.insertOne({
  _id: 1,
  name: "John Doe",
  age: 30
})
```

Ejemplo de consulta 1: Actualizar un documento existente

```
db.users.update(
  { _id: 1 },
  {
    $set: { age: 31 },
    $setOnInsert: { createdAt: new Date() }
  },
  { upsert: true }
)
```

Resultado 1

```
{
  _id: 1,
  name: "John Doe",
```

```
  age: 31
}
```

El resultado muestra que el documento se actualizó, pero el `createdAt` campo NO se agregó porque el documento ya existía. El `$setOnInsert` operador solo se aplica al insertar documentos nuevos.

Ejemplo de consulta 2: Insertar un documento nuevo (parte superior)

```
db.users.update(
  { _id: 2 },
  {
    $set: { name: "Jane Smith", age: 25 },
    $setOnInsert: { createdAt: new Date() }
  },
  { upsert: true }
)
```

Salida 2

```
{
  _id: 2,
  name: "Jane Smith",
  age: 25,
  createdAt: ISODate("2025-10-31T09:57:52.459Z")
}
}
```

Ejemplos de código

Para ver un ejemplo de código para usar el `$setOnInsert` comando, elija la pestaña del idioma que desee usar:

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

El operador de `$slice` actualización modifica una matriz limitando su tamaño. Cuando se usa con el `$push` operador, restringe el número de elementos de una matriz y mantiene solo el número especificado de elementos más recientes o más antiguos.

Parámetros

- `field`: el campo de matriz que se va a modificar.
- `count`: Número máximo de elementos que se deben conservar. Los valores positivos mantienen los primeros N elementos, los valores negativos mantienen los N últimos elementos.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el operador de `$slice` actualización con `$push` para mantener una matriz de partituras recientes de tamaño fijo.

Cree documentos de muestra

```
db.students.insertOne({
  _id: 1,
  name: "Alice",
  scores: [85, 90, 78]
});
```

Ejemplo de consulta

```
db.students.updateOne(
  { _id: 1 },
  {
    $push: {
      scores: {
        $each: [92, 88],
        $slice: -3
      }
    }
```

```
  }
)
```

Salida

```
{
  "_id" : 1,
  "name" : "Alice",
  "scores" : [ 78, 92, 88 ]
}
```

En este ejemplo, el `$slice: -3` modificador conserva solo los tres últimos elementos después de introducir nuevos valores en la matriz.

Ejemplos de código

Para ver un ejemplo de código para usar el operador de `$slice` actualización, elija la pestaña correspondiente al idioma que desee usar:

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

El modificador de `$sort` actualización ordena los elementos de la matriz cuando se usa con el `$push` operador. Organiza los elementos de la matriz en orden ascendente o descendente en función de los valores de campo especificados o de los propios elementos.

Parámetros

- `field`: El campo de matriz que se va a modificar.
- `order`: Se utiliza 1 en orden ascendente o `-1` descendente.

Ejemplo (MongoDB Shell)

En el siguiente ejemplo, se muestra el uso del `$sort` modificador con `$push` para añadir nuevas puntuaciones a las pruebas y mantenerlas ordenadas en orden descendente.

Cree documentos de muestra

```
db.students.insertOne({
  _id: 1,
  name: "Bob",
  quizzes: [
    { score: 85, date: "2024-01-15" },
    { score: 92, date: "2024-02-10" }
  ]
});
```

Ejemplo de consulta

```
db.students.updateOne(
  { _id: 1 },
  {
    $push: {
      quizzes: {
        $each: [{ score: 78, date: "2024-03-05" }],
        $sort: { score: -1 }
      }
    }
  }
)
```

Salida

```
{
  "_id" : 1,
  "name" : "Bob",
```

```
  "quizzes" : [
    { "score" : 92, "date" : "2024-02-10" },
    { "score" : 85, "date" : "2024-01-15" },
    { "score" : 78, "date" : "2024-03-05" }
  ]
}
```

Ejemplos de código

Para ver un ejemplo de código para usar el modificador de `$sort` actualización, elija la pestaña del idioma que desee usar:

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

El `$unset` operador de Amazon DocumentDB se utiliza para eliminar un campo específico de un documento. Cuando se elimina un campo utilizando`$unset`, el campo se elimina del documento y el tamaño del documento se reduce en consecuencia. Esto puede resultar útil cuando desee eliminar datos innecesarios de sus documentos.

Parámetros

- `field`: el campo que se va a eliminar del documento. Puede ser un campo único o una ruta punteada que lleve a un campo anidado.

Ejemplo (MongoDB Shell)

El siguiente ejemplo muestra cómo utilizar el `$unset` operador para eliminar el `Words` campo de un documento de la `example` colección.

Cree documentos de muestra

```
db.example.insert({
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    },
    "Words": 2482
})
```

Ejemplo de consulta

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

Salida

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

En este ejemplo, el `$unset` operador se utiliza para eliminar el `Words` campo del documento con `DocName` el valor «Documento 1". El documento resultante ya no contiene el `Words` campo.

Ejemplos de código

Para ver un ejemplo de código para usar el `$unset` comando, elija la pestaña correspondiente al idioma que desee usar:

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()
```

Administración de clústeres e instancias de Amazon DocumentDB

En estas secciones se describen los distintos componentes y sus tareas relacionadas para administrar la implementación de Amazon DocumentDB (compatible con MongoDB).

Temas

- Descripción general de las tareas operativas de Amazon DocumentDB
- Información general de los clústeres globales de Amazon DocumentDB
- Administración de clústeres de Amazon DocumentDB
- Administración de instancias de Amazon DocumentDB
- Administración de grupos de subredes de Amazon DocumentDB
- Alta disponibilidad y replicación de Amazon DocumentDB
- Administración de eventos de Amazon DocumentDB
- Administración de los grupos de parámetros de clúster de Amazon DocumentDB
- Descripción de los puntos de conexión de Amazon DocumentDB
- Descripción de nombres de recurso de Amazon (ARN) en Amazon DocumentDB
- Etiquetado de recursos de Amazon DocumentDB
- Mantenimiento de Amazon DocumentDB
- Descripción de los roles vinculados a servicios

Descripción general de las tareas operativas de Amazon DocumentDB

En esta sección se explican las tareas operativas de un clúster de Amazon DocumentDB y cómo llevar a cabo estas tareas mediante la AWS CLI.

Temas

- Agregación de una réplica a un clúster de Amazon DocumentDB
- Descripción de clústeres e instancias

- Creación de una instantánea de un clúster
- Restauración a partir de una instantánea
- Eliminación de una instancia de un clúster
- Eliminación de un clúster

Agregación de una réplica a un clúster de Amazon DocumentDB

Después de crear la instancia principal de su clúster de Amazon DocumentDB, puede añadir una o varias réplicas. Una réplica es una instancia de solo lectura que tiene dos finalidades:

- Escalabilidad: si tiene gran cantidad de clientes que deben obtener acceso simultáneamente, puede agregar más réplicas para escalar la lectura.
- Alta disponibilidad: si se produce un error en la instancia primaria, Amazon DocumentDB automáticamente realiza una conmutación por error a una instancia de réplica y la designa como nuevo nodo primario. Si se produce un error en una réplica, otras instancias del clúster pueden seguir atendiendo las solicitudes hasta que se consigue recuperar el nodo defectuoso.

Cada clúster de Amazon DocumentDB puede admitir hasta 15 réplicas.

> **Note**
>
> Para disfrutar de la máxima tolerancia a errores, debe implementar réplicas en zonas de disponibilidad distintas. Esta configuración garantiza que el clúster de Amazon DocumentDB continúe funcionando aunque una zona de disponibilidad completa deje de estar disponible.

El siguiente AWS CLI ejemplo muestra cómo agregar una nueva réplica. El parámetro `--availability-zone` coloca la réplica en la zona de disponibilidad especificada

```
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
```

Descripción de clústeres e instancias

El siguiente AWS CLI ejemplo muestra todos los clústeres de Amazon DocumentDB de una región. Para ciertas características de administración, como la administración del ciclo de vida de clúster y de instancia, Amazon DocumentDB aprovecha la tecnología operativa que se comparte con Amazon RDS. El parámetro de filtro `filterName=engine,Values=docdb` devuelve solo clústeres de Amazon DocumentDB.

Para obtener más información sobre cómo describir y modificar clústeres, consulte Ciclo de vida del clúster de Amazon DocumentDB.

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "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",
            ...
        }
    ]
}
```

El siguiente AWS CLI ejemplo muestra las instancias de un clúster de Amazon DocumentDB. Para obtener más información sobre cómo describir y modificar clústeres, consulte Ciclo de vida de instancia de Amazon DocumentDB.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterMembers]'
```

El resultado es similar al siguiente. En este resultado hay dos instancias. La instancia principal es `sample-instance-1` (`"IsClusterWriter": true`). También hay una instancia de réplica, `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
            }
        ]
    ]
]
```

Creación de una instantánea de un clúster

Una instantánea de clúster es una copia de seguridad completa de los datos del clúster de Amazon DocumentDB. Cuando la instantánea se está creando, Amazon DocumentDB lee los datos directamente del volumen del clúster. Por este motivo, puede crear una instantánea aunque el clúster no tenga instancias en ejecución en ese momento. La cantidad de tiempo que tarda en crearse una instantánea depende del tamaño del volumen del clúster.

Amazon DocumentDB admite copias de seguridad automáticas, que se realizan a diario durante el período de copia de seguridad preferido, es decir, un período de 30 minutos durante el día. El siguiente AWS CLI ejemplo muestra cómo ver la ventana de respaldo de su clúster:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].PreferredBackupWindow'
```

El resultado muestra el periodo de copia de seguridad (en UTC):

```
[
    "00:18-00:48"
]
```

Puede definir el periodo de copia de seguridad cuando cree el clúster de Amazon DocumentDB. También puede cambiar el periodo de copia de seguridad, tal y como se muestra en el siguiente ejemplo: Si no define un periodo de copia de seguridad, Amazon DocumentDB asigna uno automáticamente a su clúster.

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --preferred-backup-window "02:00-02:30"
```

Además de copias de seguridad automáticas, puede crear manualmente una instantánea del clúster en cualquier momento. Cuando cree la instantánea, especifique el clúster del que desea hacer

una copia de seguridad y un nombre único para la instantánea, de forma que pueda realizar la restauración desde ella en otro momento.

El siguiente AWS CLI ejemplo muestra cómo crear una instantánea de sus datos.

```
aws docdb create-db-cluster-snapshot \
    --db-cluster-identifier sample-cluster \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Restauración a partir de una instantánea

Puede restaurar una instantánea de clúster en un nuevo clúster de Amazon DocumentDB. Para ello, proporcione el nombre de la instantánea y el nombre de un nuevo clúster. No puede restaurar desde una instantánea a un clúster existente; en su lugar, Amazon DocumentDB crea un nuevo clúster al restaurar y, a continuación, lo rellena con los datos de su instantánea.

El siguiente ejemplo muestra todas las instantáneas de un clúster `sample-cluster`.

```
aws docdb describe-db-cluster-snapshots \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusterSnapshots[*].[DBClusterSnapshotIdentifier,SnapshotType,Status]'
```

El resultado es similar al siguiente. Una instantánea manual es aquella que se crea manualmente, mientras que una instantánea automatizada la crea Amazon DocumentDB dentro del periodo de copia de seguridad del clúster.

```
[
        "sample-cluster-snapshot",
        "manual",
        "available"
    ],
    [
        "rds:sample-cluster",
        "automated",
        "available"
    ]
]
```

En el siguiente ejemplo se muestra cómo restaurar un clúster de Amazon DocumentDB a partir de una instantánea.

```
aws docdb restore-db-cluster-from-snapshot \
    --engine docdb \
    --db-cluster-identifier new-sample-cluster \
    --snapshot-identifier sample-cluster-snapshot
```

El nuevo clúster no tiene instancias asociadas; por lo tanto, si desea interactuar con el clúster, debe añadirle una instancia.

```
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
```

Puede utilizar las siguientes AWS CLI operaciones para supervisar el progreso de la creación de clústeres e instancias. Cuando el clúster y la instancia tengan estados disponibles, podrá conectarse al punto de conexión del nuevo clúster y obtener acceso a sus datos.

```
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]'
```

Eliminación de una instancia de un clúster

Amazon DocumentDB almacena todos los datos en el volumen del clúster. Los datos se conservan en ese volumen de clúster, incluso si elimina todas las instancias del clúster. Si necesita obtener acceso a los datos de nuevo, puede añadir una instancia a la clúster en cualquier momento y reanudar el trabajo donde lo dejó.

En el siguiente ejemplo se muestra cómo eliminar una instancia de su clúster de Amazon DocumentDB.

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-instance
```

Eliminación de un clúster

Para poder eliminar un clúster de Amazon DocumentDB, primero debe eliminar todas sus instancias. El siguiente AWS CLI ejemplo devuelve información sobre las instancias de un clúster. Si esta operación devuelve algún identificador de instancia, tendrá que eliminar cada una de esas instancias. Para obtener más información, consulte Eliminación de una instancia de un clúster.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterMembers[*].DBInstanceIdentifier'
```

Cuando no queden más instancias, puede eliminar el clúster. En ese momento, debe elegir una de las siguientes opciones:

- Cree una instantánea final: capture todos los datos del clúster en una instantánea para poder volver a crear una nueva instancia con esos datos más adelante. El siguiente ejemplo le muestra cómo hacerlo:

  ```
  aws docdb delete-db-cluster \
      --db-cluster-identifier sample-cluster \
      --final-db-snapshot-identifier sample-cluster-snapshot
  ```

- Omita la instantánea final: descarte permanentemente todos los datos del clúster. Esta operación no se puede deshacer. El siguiente ejemplo le muestra cómo hacerlo:

  ```
  aws docdb delete-db-cluster \
      --db-cluster-identifier sample-cluster \
      --skip-final-snapshot
  ```

Información general de los clústeres globales de Amazon DocumentDB

¿Qué es un clúster global?

Un clúster global consta de una región principal y hasta 10 regiones secundarias de solo lectura. Las operaciones de escritura se envían directamente al clúster principal de la región principal y Amazon DocumentDB replica automáticamente los datos en las regiones secundarias mediante una infraestructura dedicada. La latencia suele ser inferior a un segundo.

¿Para qué sirven los clústeres globales?

- Recuperación ante Region-wide interrupciones: en caso de que se produzca una Region-wide interrupción, puede convertir uno de los clústeres secundarios en clúster principal en cuestión de minutos, con un objetivo de tiempo de recuperación (RTO) típico de menos de un minuto. El objetivo de punto de recuperación (RPO) suele medirse en segundos, pero depende del desfase existente en la red en el momento del fallo.
- Lecturas globales con latencia local: si tiene oficinas en todo el mundo, puede usar un clúster global para mantener actualizadas sus principales fuentes de información en la región principal. Las oficinas en sus otras regiones pueden acceder a la información en su propia región, con latencia local.
- Clústeres secundarios escalables: puedes escalar tus clústeres secundarios añadiendo más instancias de solo lectura a una región secundaria. El clúster secundario es de solo lectura, por lo que puede admitir hasta 16 instancias de réplica de solo lectura en lugar del límite habitual de 15 para un solo clúster.
- Replicación rápida de clústeres primarios a secundarios - La replicación realizada por un clúster global tiene poco impacto en el performance en el clúster primario. Los recursos de las instancias de bases de datos están totalmente dedicados a servir a las cargas de trabajo de lectura y escritura de la aplicación.
- Flujos de cambios: puedes leer los flujos de cambios de un clúster secundario en clústeres globales, lo que reduce la carga del clúster principal y te permite escalar las cargas de trabajo de lectura de las regiones secundarias.

¿Cuáles son las limitaciones actuales de los clústeres globales?

- Los clústeres globales no son compatibles con Amazon DocumentDB v3.6.
- Los clústeres globales son compatibles con todos los tipos de instancias, excepto db.t3, db.t4g y db.r4.
- Los clústeres globales no están disponibles en las siguientes regiones: América del Sur (São Paulo), Europa (Milán), China (Pekín) y China (Ningxia).
- La conmutación y la conmutación por error global no se admiten cuando las regiones utilizan diferentes versiones de motor. Se admite la conmutación por error manual cuando existe una discordancia entre las versiones del motor.
- Solo el clúster primario realiza operaciones de escritura. Los clientes que realizan operaciones de escritura se conectan al punto de conexión del clúster primario.

- Puedes tener un máximo de 10 regiones secundarias y una región principal para tu clúster.
- No se puede detener un clúster secundario. No se puede detener un clúster primario si tiene clústeres secundarios asociados. Solo se puede detener un clúster regional que no tenga clústeres secundarios.
- Las réplicas conectadas al clúster secundario pueden reiniciarse en determinadas circunstancias. Si la instancia de la región principal se reinicia o se conmuta por error, las réplicas de la región secundaria también se reinician. El clúster no estará disponible hasta que todas las réplicas estén nuevamente sincronizadas con la instancia del escritor del clúster de base de datos principal. Este es el comportamiento esperado. Asegúrese de comprender los impactos en la base de datos global antes de realizar cambios en el clúster primario.

Temas

- Guía de inicio rápido: clústeres globales
- Administración de un clúster global de Amazon DocumentDB
- Conexión a un clúster global de Amazon DocumentDB
- Supervisión de clústeres globales de Amazon DocumentDB
- Recuperación ante desastres y clústeres globales de Amazon DocumentDB

Guía de inicio rápido: clústeres globales

Temas

- Configuración
- Creación de un clúster global de Amazon DocumentDB
- Agregar un Región de AWS a un clúster global de Amazon DocumentDB
- Uso de una instantánea para el clúster global de Amazon DocumentDB

Configuración

El clúster global de Amazon DocumentDB abarca al menos dos. Regiones de AWS La región principal admite un clúster que tiene una instancia principal (de escritura) y hasta 15 instancias de réplica, mientras que la región secundaria ejecuta un clúster de solo lectura compuesto en su totalidad por un máximo de 16 instancias de réplica. Un clúster global puede tener hasta 10 regiones secundarias. La tabla muestra el máximo de clústeres, instancias y réplicas permitidos en clúster global.

Description (Descripción)	Primaria Región de AWS	Secundario Región de AWS
Clústeres	1	10 (máximo)
Instancias de escritor	1	0
Read-only instancias (réplicas de Amazon DocumentDB), por clúster	15 (máx.)	16 (total)
Read-only instancias (máximo permitido, dado el número real de regiones secundarias)	15 - s	s = número total de secundarias Regiones de AWS

Los clústeres tienen los siguientes requisitos específicos:

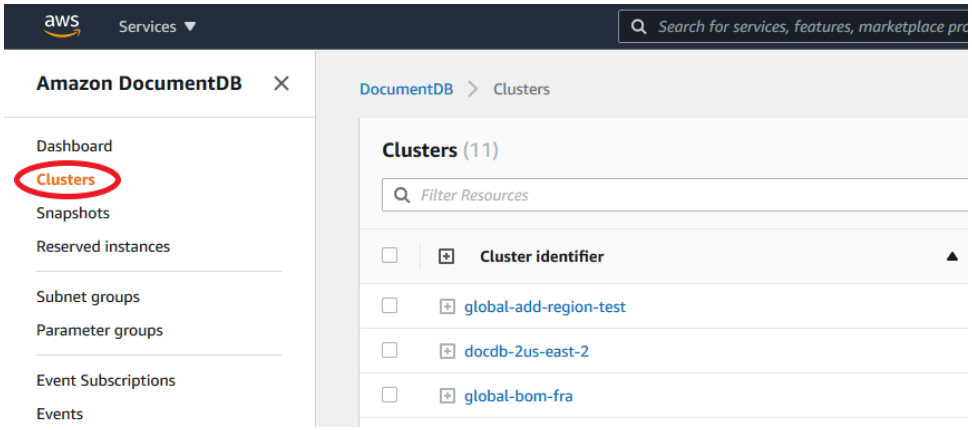
- Requisitos de clase de instancia de base de datos: solo puede usar las clases de instancia `db.r5` y `db.r6g`.
- Región de AWS requisitos: el clúster principal debe estar en una región y al menos un clúster secundario debe estar en una región diferente de la misma cuenta. Puedes crear hasta 10 clústeres secundarios (de solo lectura) y cada uno debe estar en una región diferente. En otras palabras, no puede haber dos clústeres en la misma región.
- Requisitos de nomenclatura: los nombres que elija para cada uno de los clústeres deben ser únicos, en todas las regiones. No puede usar el mismo nombre para diferentes clústeres aunque estén en diferentes regiones.

Creación de un clúster global de Amazon DocumentDB

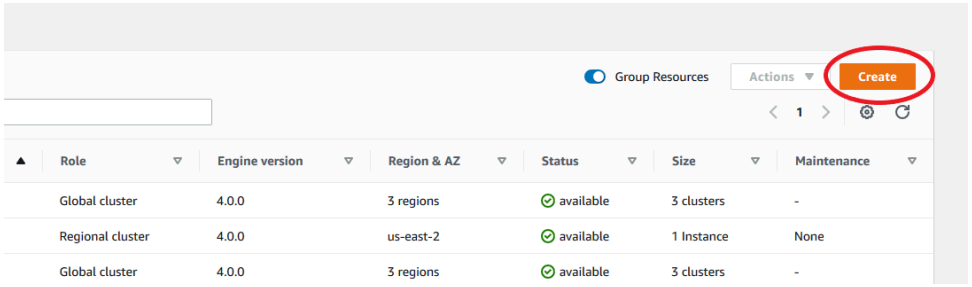
¿Está preparado para compilar su primer clúster global? En esta sección, explicaremos cómo crear un clúster global completamente nuevo con nuevos clústeres e instancias de bases de datos, mediante las siguientes instrucciones Consola de administración de AWS o mediante AWS CLI las siguientes instrucciones.

Uso de Consola de administración de AWS

1. En Consola de administración de AWS, navegue hasta Amazon DocumentDB.
2. Cuando llegue a la consola de Amazon DocumentDB, elija Clústeres.

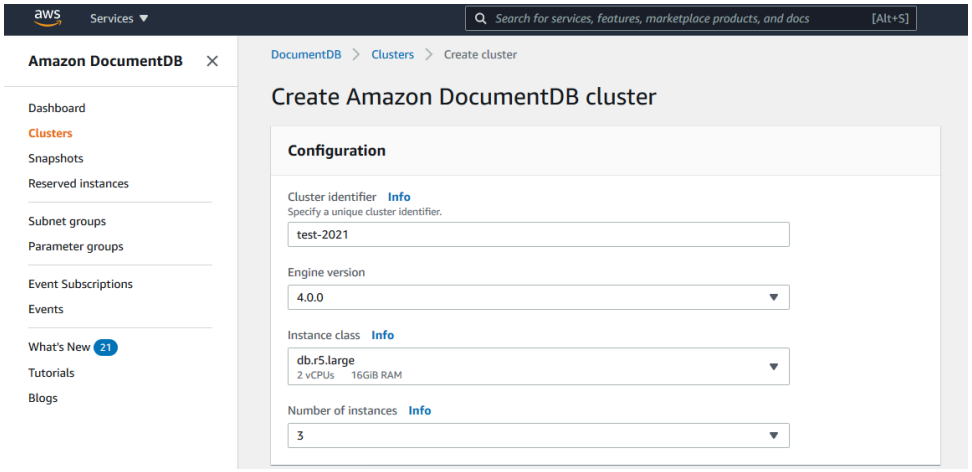


3. Seleccione Crear.

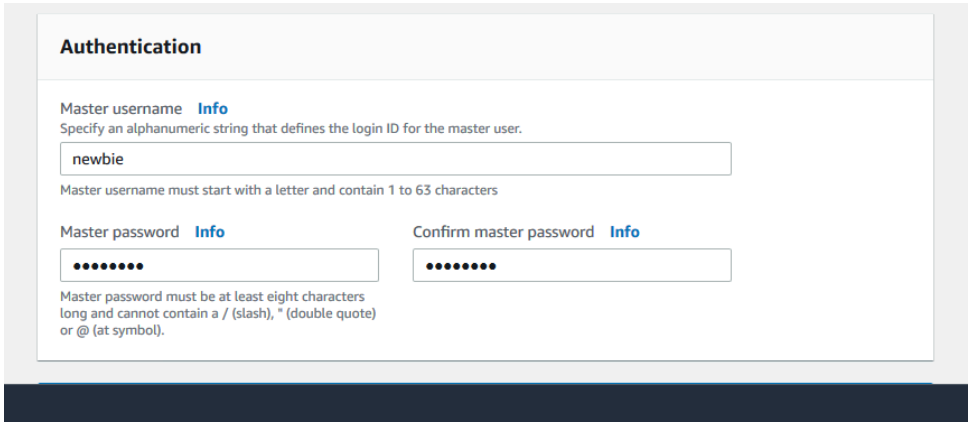


4. Rellene la sección Configuración del formulario Crear un clúster de Amazon DocumentDB según corresponda:

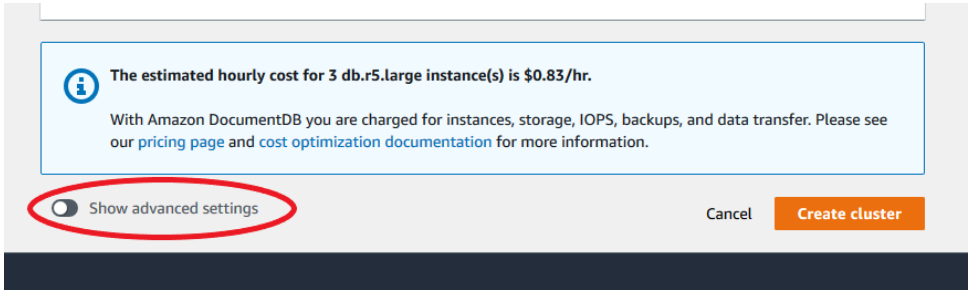
   - Identificador de clúster puede escribir un identificador único para esta instancia o permitir que Amazon DocumentDB proporcione el identificador de la instancia basándose en el identificador del clúster.
   - Elija la versión del motor 4.0.0
   - Clase de instancia: elija db.r5.large
   - Número de instancias: elija 3.



5. En la sección Autenticación, introduzca un nombre de usuario maestro y una contraseña maestra.

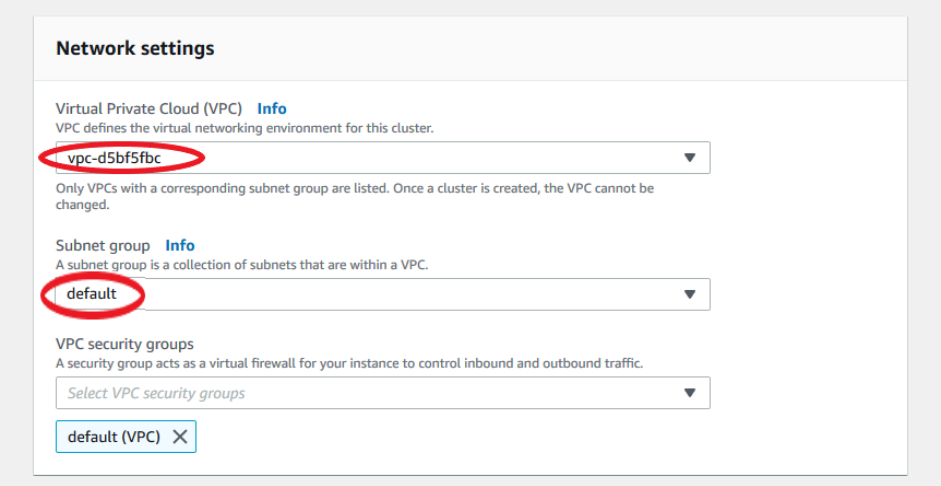


6. Seleccione Mostrar configuración avanzada.

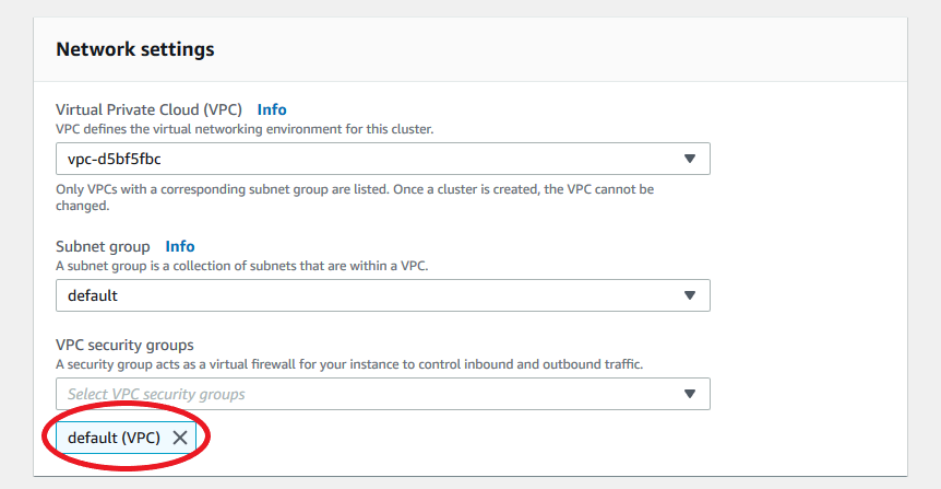


7. En la sección Configuración de red:

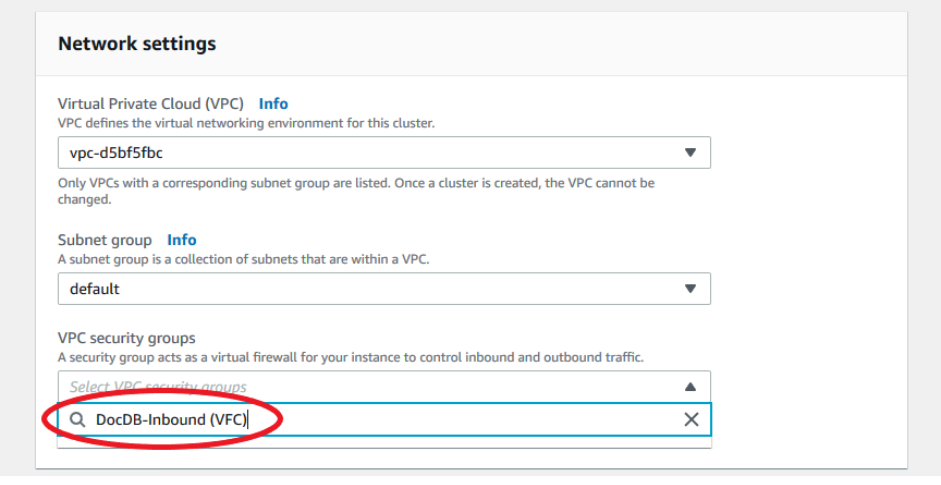
   - Mantenga las opciones predeterminadas para la Nube privada virtual (VPC) y el Grupo de subredes.



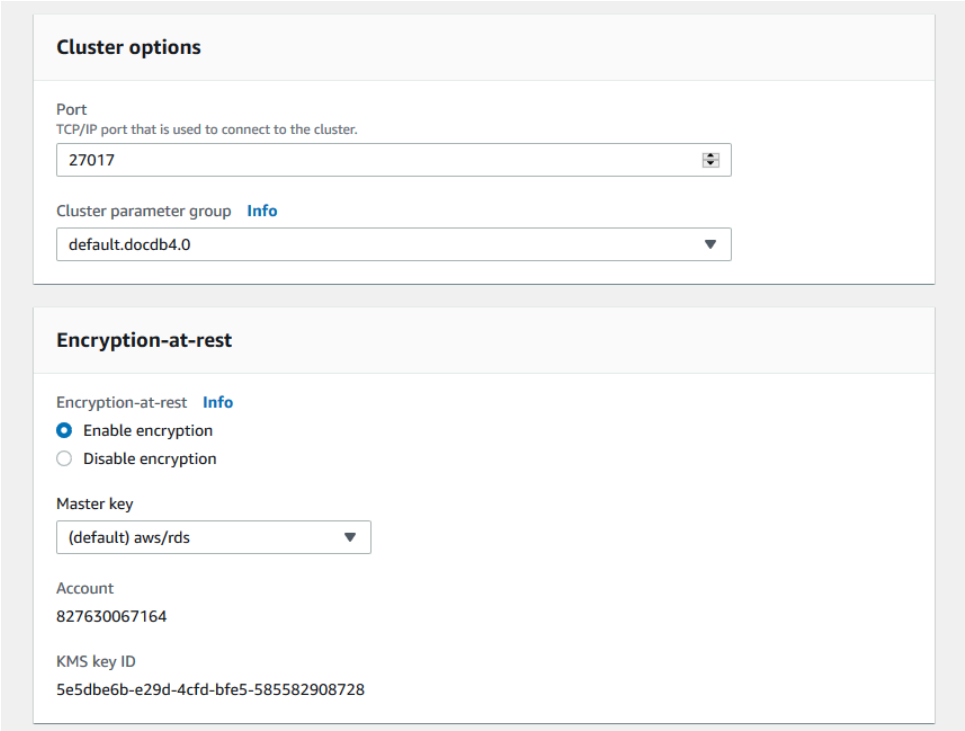
- Para los grupos de seguridad de VPC, la (VPC) predeterminada ya debería estar añadida.



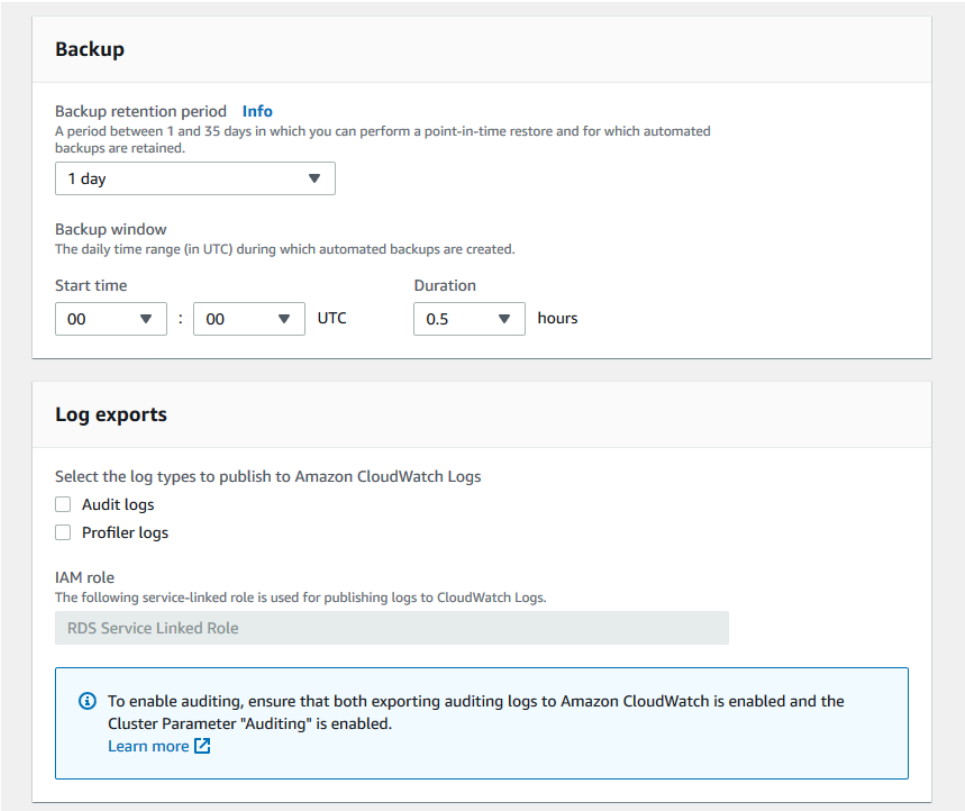
- Escriba DocDB en el campo de grupos de seguridad de la VPC y seleccione DocDB-Inbound (VFC).



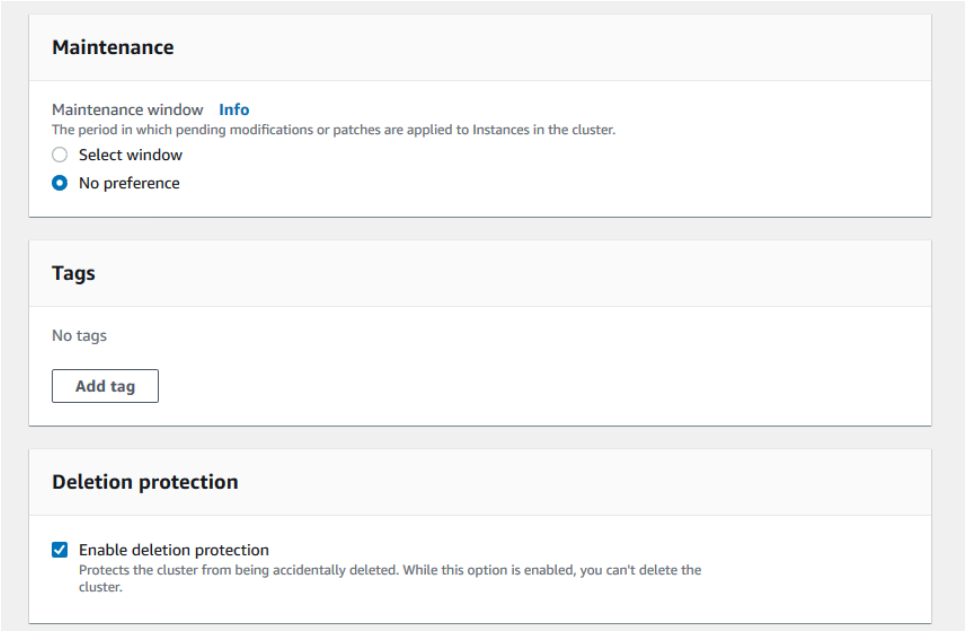
8. Para las opciones de clúster y Encryption-at-rest, déjelas en las selecciones predeterminadas.



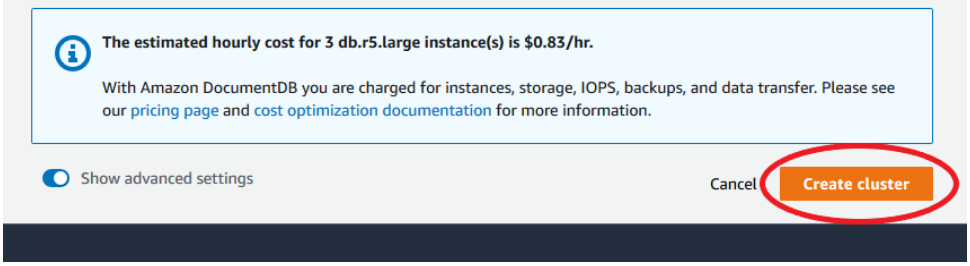
9. Para Copia de seguridad y Exportaciones de registros, deje las selecciones predeterminadas.



10. Para Mantenimiento, Etiquetas y Protección contra eliminación, deje las selecciones predeterminadas.



11. Ahora elige Crear clúster.



Uso de AWS CLI

Para crear un clúster regional de Amazon DocumentDB, llame al clúster AWS CLI create-global-cluster. El siguiente AWS CLI comando crea un clúster de Amazon DocumentDB denominado. `global-cluster-id` Para obtener más información sobre la protección contra eliminación, consulte Eliminación de un clúster de Amazon DocumentDB.

Además, `--engine-version` es un parámetro opcional que utiliza de forma predeterminada la última versión principal del motor. La versión predeterminada actual del motor es `5.0.0` (nota: Amazon DocumentDB 8.0 está disponible, pero debe especificarse explícitamente como`8.0.0`). Cuando se publican nuevas versiones principales del motor, la versión predeterminada del motor para `--engine-version` se actualiza para reflejar la última versión principal del motor. Como resultado, en el caso de las cargas de trabajo de producción, y especialmente de las que dependen de la creación de scripts, la automatización o CloudFormation las plantillas, se especifica explícitamente la `--engine-version` versión principal deseada.

Si no se especifica un `db-subnet-group-name` o un `vpc-security-group-id`, Amazon DocumentDB utilizará el grupo de subredes y el grupo de seguridad de Amazon VPC predeterminados para la región determinada.

Para utilizar el ejemplo siguiente, sustituya *`user input placeholder`* con su propia información.

Para 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
```

Para 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
```

La salida de esta operación será similar a lo que se indica a continuación (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 creación del clúster puede tardar varios minutos. Puedes usar el Consola de administración de AWS o AWS CLI para supervisar el estado de tu clúster. Para obtener más información, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

> **Important**
>
> Cuando lo utiliza AWS CLI para crear un clúster regional de Amazon DocumentDB, no se crea ninguna instancia. Por lo tanto, tendrá que crear de forma explícita una instancia principal y las instancias de réplica que necesite. Puede usar la consola o AWS CLI crear las instancias. Para obtener más información, consulte Agregación de una instancia de Amazon DocumentDB a un clúster y cree DBCluster en la referencia de la API de Amazon DocumentDB.

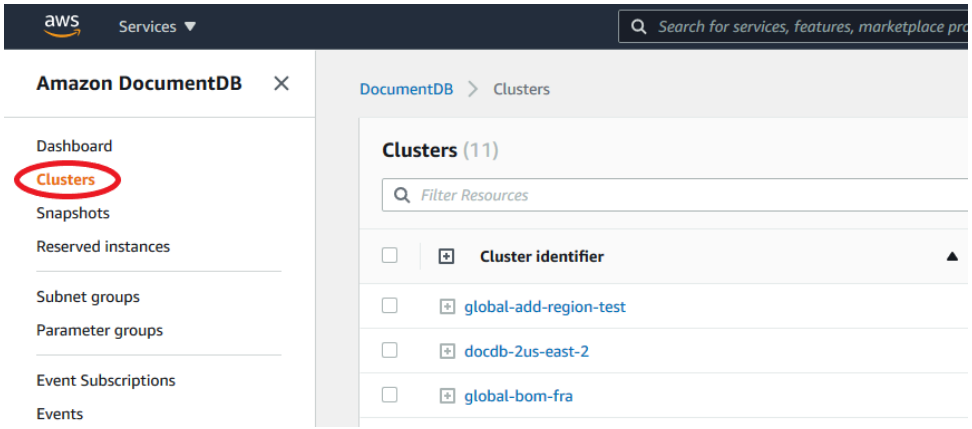
Una vez que su clúster regional esté disponible, puede añadir un clúster secundario en otra región con las siguientes instrucciones: Agregar un Región de AWS a un clúster global de Amazon DocumentDB. Cuando añada una región, su clúster regional se convierte en su clúster principal y tiene un nuevo clúster secundario en la región que ha elegido.

Agregar un Región de AWS a un clúster global de Amazon DocumentDB

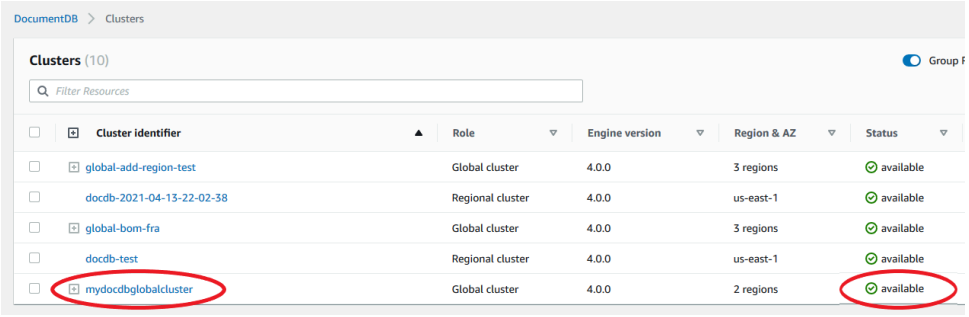
Un clúster global necesita al menos un clúster secundario en una región diferente a la del clúster principal, y puede añadir hasta 10 clústeres secundarios. Para cada clúster secundario que agregue, deberá reducir el número de réplicas permitidas al clúster principal en una. Por ejemplo, si tu clúster global tiene 10 regiones secundarias, tu clúster principal solo puede tener 5 réplicas (en lugar de 15). Para obtener más información, consulte Configuración.

Uso de Consola de administración de AWS

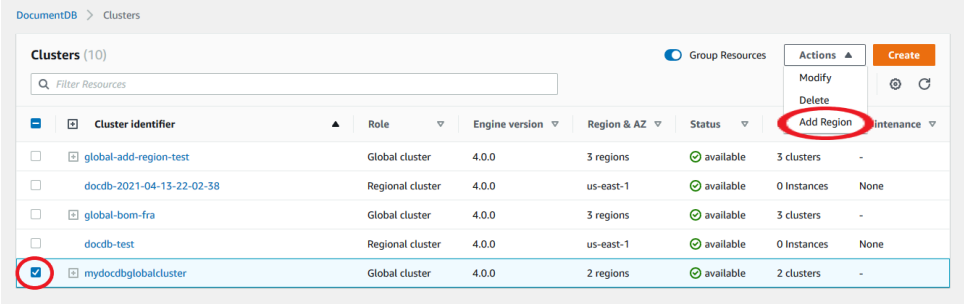
1. Inicie sesión en la consola de Amazon DocumentDB Consola de administración de AWS y ábrala.
2. En el panel de navegación, seleccione Clusters (Clústeres).



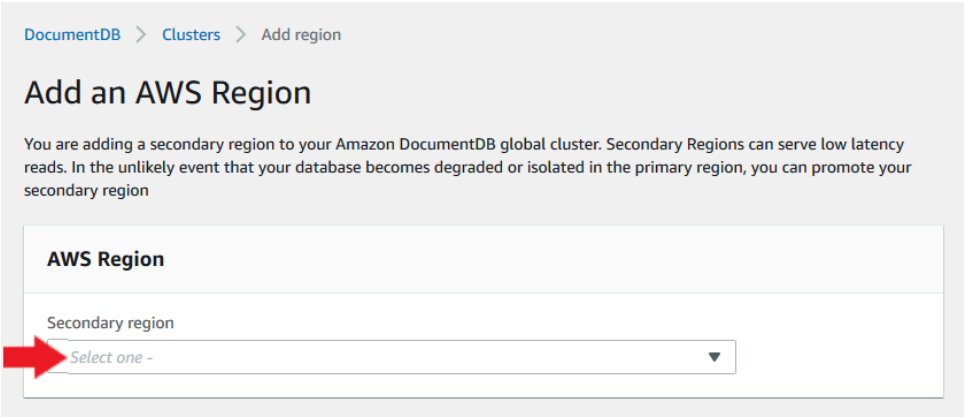
3. Elija el clúster al que desearía agregar un clúster secundario. Asegúrese de que el clúster es `Available`.



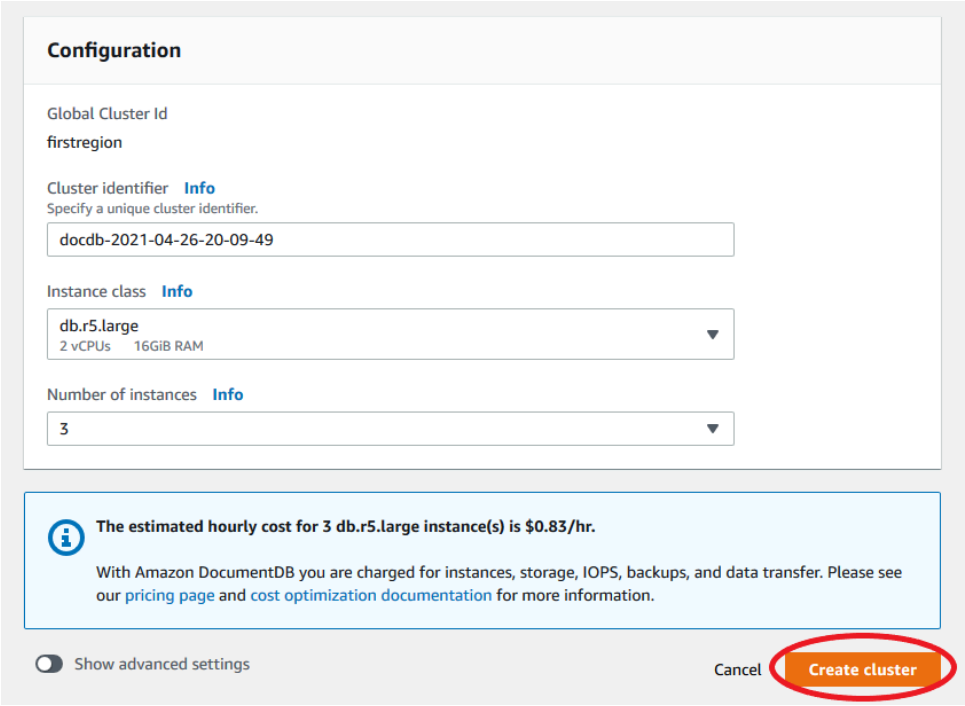
4. Seleccione el menú desplegable Acciones y, a continuación, seleccione Añadir región.



5. En la página Añadir una Región de AWS, seleccione la región secundaria. No puede elegir una región que ya tenga un clúster secundario para el mismo clúster global. Además, no puede ser la misma región que el clúster principal. Si es la primera región que va a añadir, también tendrá que especificar un identificador de clúster global de su elección.



6. Complete los campos restantes para el clúster secundario en la nueva región, luego seleccione Crear clúster. Después de terminar de añadir la región, puede verla en la lista de Clústeres en la Consola de administración de AWS.



Uso de AWS CLI

- Utilice el comando `create-db-cluster` CLI con el nombre (`--global-cluster-identifier`) del clúster global. Para otros parámetros, haga lo siguiente:
  - `--region`En caso contrario, elija una región Región de AWS diferente a la de su región principal.
  - Elija valores específicos para los parámetros `--engine` y `--engine-version`.
  - En el caso de un clúster cifrado, especifique su clúster principal Región de AWS como el `--source-region` de cifrado.

En el ejemplo siguiente se crea un nuevo clúster de Amazon DocumentDB y se adjunta al clúster global como clúster secundario de solo lectura. En el último paso, se agrega una instancia al nuevo clúster.

Para utilizar el ejemplo siguiente, sustituya *`user input placeholder`* con su propia información.

Para 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
```

Para 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
```

Uso de una instantánea para el clúster global de Amazon DocumentDB

Puede restaurar una instantánea de un clúster de Amazon DocumentDB para utilizarla como punto de partida del clúster global. Para ello, debe restaurar la instantánea y crear un nuevo clúster. Servirá como el clúster principal de su clúster global. A continuación, añada otra región al clúster restaurado, convirtiéndolo así en clúster global.

Administración de un clúster global de Amazon DocumentDB

Puede realizar la mayor parte de las operaciones de administración en los clústeres individuales que componen un clúster global. Cuando selecciona Recursos relacionados con grupos en la página

Clústeres de la consola, verá el clúster primario y los clústeres secundarios agrupados bajo el objeto de clúster global asociado.

La pestaña Configuración de un clúster global muestra Regiones de AWS dónde se ejecutan los clústeres, la versión y el identificador del clúster global.

Temas

- Modificación de un clúster global de Amazon DocumentDB
- Modificación de los parámetros de un clúster global de Amazon DocumentDB
- Eliminación de un clúster global de Amazon DocumentDB
- Eliminación de un clúster global de Amazon DocumentDB
- Creación de un clúster independiente de Amazon DocumentDB en una región secundaria

Modificación de un clúster global de Amazon DocumentDB

La página Clústeres de la Consola de administración de AWS lista muestra todos los clústeres globales y muestra el clúster principal y los clústeres secundarios de cada uno. El clúster global tiene sus propias opciones de configuración. En concreto, tiene regiones asociadas a sus clústeres principales y secundarios.

Al realizar cambios en el clúster global, tiene la oportunidad de cancelar los cambios.

Al seleccionar Continue (Continuar), confirma los cambios.

Modificación de los parámetros de un clúster global de Amazon DocumentDB

Puede configurar los grupos de parámetros de clúster independientemente para cada clúster dentro del clúster global. La mayoría de parámetros funcionan igual que para otros tipos de clústeres de Amazon DocumentDB. Mantén la misma configuración en todos los clústeres de una base de datos global para evitar cambios de comportamiento inesperados si promocionas un clúster secundario para que pase a ser el principal.

Por ejemplo, utilice la misma configuración de zonas horarias y conjuntos de caracteres para evitar un comportamiento incoherente si un clúster diferente asume la función del clúster principal.

Eliminación de un clúster global de Amazon DocumentDB

Hay varias situaciones en las que es posible que desee eliminar clústeres de su clúster global. Por ejemplo, es posible que desee quitar un clúster de un clúster global si el clúster principal se

degrada o se aísla. A continuación, se convierte en un clúster provisionado independiente que podría utilizarse para crear un nuevo clúster global. Para obtener más información, consulte Ejecución de una conmutación por error manual para un clúster global de Amazon DocumentDB.

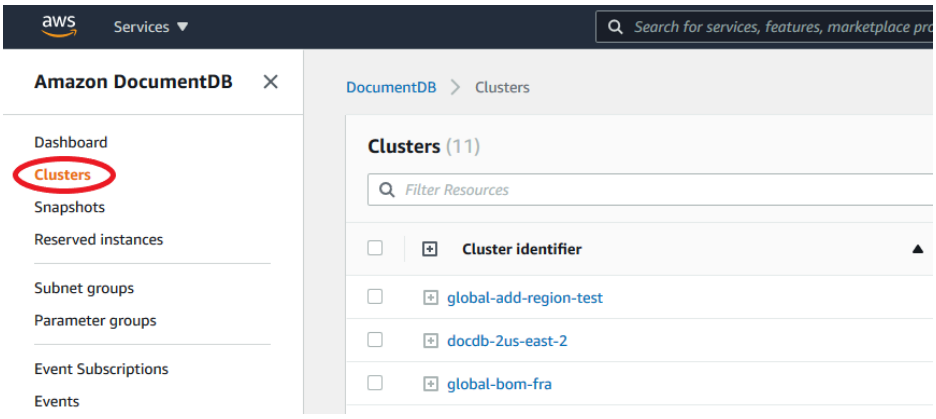
También puede querer quitar clústeres porque desea eliminar un clúster global que ya no necesite. No puede eliminar el clúster global hasta después de eliminar (desasociar) todos los clústeres asociados y deje el principal para lo último. Para obtener más información, consulte Eliminación de un clúster global de Amazon DocumentDB.

> **Note**
>
> Cuando un clúster se desasocia del clúster global, ya no se sincroniza con el principal. Se convierte en un clúster aprovisionado independiente con todas read/write las capacidades. Además, ya no está visible en la consola de Amazon DocumentDB. Solo está visible cuando se selecciona la región en la consola en la que estaba ubicado el clúster.

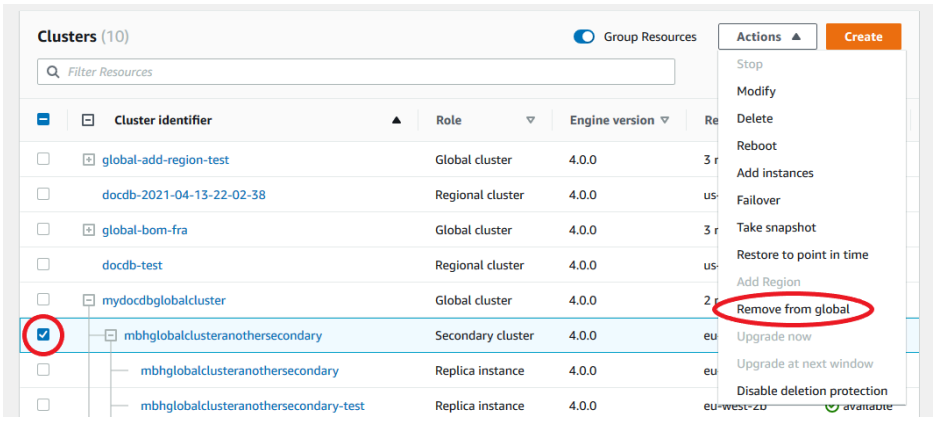
Puedes eliminar clústeres de tu clúster global mediante la API Consola de administración de AWS AWS CLI, la API o la API de RDS.

Using the Consola de administración de AWS

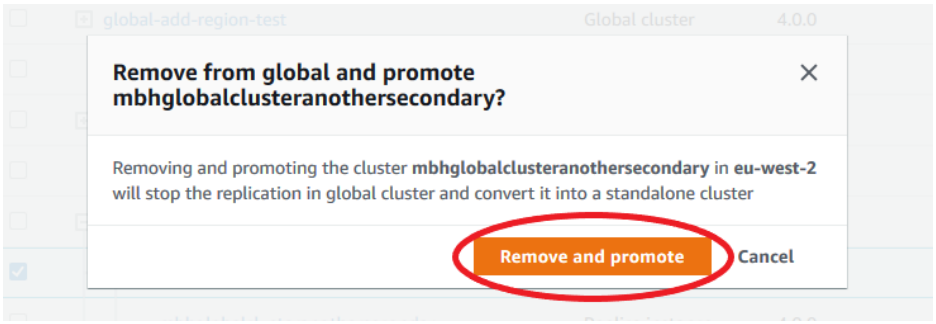
1. Inicie sesión en la consola de Amazon DocumentDB Consola de administración de AWS y navegue hasta ella.
2. En el panel de navegación izquierdo, elija Clústeres.



3. Amplíe el clúster global para que pueda ver todos los clústeres secundarios. Seleccione los clústeres secundarios que desee eliminar. Selecciona Acciones y, en el menú desplegable, selecciona Eliminar del mapa global.



4. Abra un mensaje para confirmar que desea separar el secundario del clúster global. Elija Eliminar y promover para quitar el clúster del clúster global.



El clúster ya no sirve como secundario y ya no está sincronizado con el clúster principal. Se trata de un clúster independiente con plena read/write capacidad.

Tras eliminar o borrar todos los clústeres secundarios, podrá eliminar el clúster principal del mismo modo. No puede separar o quitar el clúster principal de un clúster global hasta después de quitar todos los clústeres secundarios. El clúster global puede permanecer en la lista de clústeres, sin regiones ni zonas de disponibilidad. Puede eliminar si ya no desea utilizar este clúster global.

Using the AWS CLI

Para eliminar un clúster de un clúster global, ejecute el comando CLI `remove-from-global-cluster` con los siguientes parámetros:

- `--global-cluster-identifier`: nombre (identificador) del clúster global.
- `--db-cluster-identifier`: nombre de cada clúster que se va a quitar del clúster global.

Los siguientes comandos eliminan un clúster secundario y, después, el clúster primario de un clúster global.

Para 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
```

Repite el `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` comando para cada región secundaria de tu clúster global.

Para 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
```

Repite el `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` comando para cada región secundaria de tu clúster global.

Eliminación de un clúster global de Amazon DocumentDB

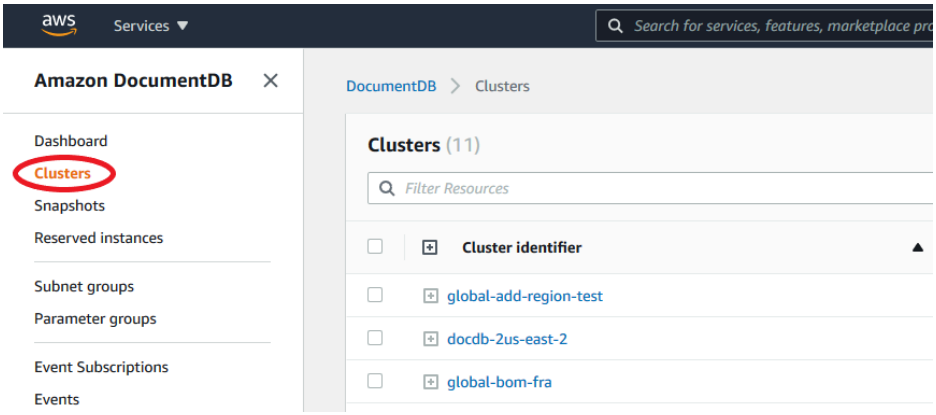
Para completar la eliminación de un clúster global, siga estos pasos:

- Elimine todas las regiones secundarias del clúster global. Cada clúster se convierte en un clúster independiente. Consulta la sección anterior,Eliminación de un clúster global de Amazon DocumentDB.
- En cada clúster independiente, elimine todas las réplicas.
- Elimine el clúster secundario del clúster global. Esto se convierte en un clúster independiente.

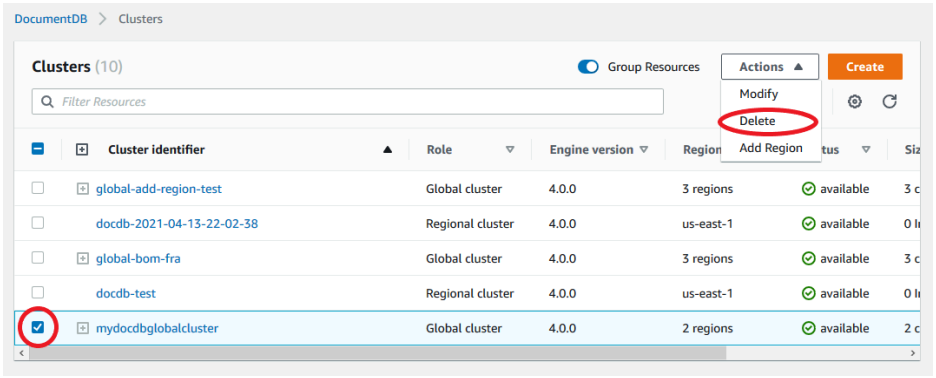
- Desde el clúster principal, primero elimine todas las réplicas y, a continuación, elimine la instancia primaria. La eliminación de la instancia de escritor del clúster recién independiente también normalmente elimina el clúster y el clúster global.

Using the Consola de administración de AWS

1. Inicie sesión en la consola de Amazon DocumentDB Consola de administración de AWS y navegue hasta ella.
2. Elija Clústeres y busque el clúster global que desea eliminar.



3. Con el clúster global seleccionado, selecciona Eliminar en el menú Acciones.



Confirme que todos los demás clústeres se han borrado del clúster global. El clúster global debe mostrar cero regiones y zonas de disponibilidad y un tamaño de cero clústeres. Si el clúster global contiene clústeres, no puede eliminarla. Primero tendrá que seguir las instrucciones del paso anterior, Eliminación de un clúster global de Amazon DocumentDB.

Using the AWS CLI

Para eliminar un clúster global, ejecute el comando de la `delete-global-cluster` CLI con el nombre Región de AWS y el identificador del clúster global, como se muestra en el siguiente ejemplo.

Para Linux, macOS o Unix:

```
aws docdb --region primary_region delete-global-cluster \
   --global-cluster-identifier global_cluster_id
```

Para Windows:

```
aws docdb --region primary_region delete-global-cluster ^
   --global-cluster-identifier global_cluster_id
```

Creación de un clúster independiente de Amazon DocumentDB en una región secundaria

Si bien un clúster global de Amazon DocumentDB requiere al menos un clúster secundario en un Región de AWS lugar diferente al principal, puede usar una configuración independiente para el clúster secundario. Un clúster secundario de Amazon DocumentDB sin periféricos es uno sin una instancia. Este tipo de configuración puede reducir los gastos para un clúster global. En un clúster de Amazon DocumentDB, se desacoplan la informática y el almacenamiento. Sin la instancia, no se le cobrará por la informática, solo por almacenamiento. Si está configurado correctamente, el volumen de almacenamiento de un clúster secundario sin periféricos se mantiene sincronizado con el clúster principal.

Agregue el clúster secundario como lo hace normalmente al crear un clúster global de Amazon DocumentDB. Sin embargo, después de que el clúster principal comience la reproducción en el secundario, se elimina la instancia de solo lectura del clúster secundario. Este clúster secundario ahora se considera «independiente» porque ya no tiene una instancia. Sin embargo, el volumen de almacenamiento se mantiene sincronizado con el clúster principal de Amazon DocumentDB.

> **Important**
>
> Usa clústeres independientes solo si puedes tolerar Region-wide errores durante 15 minutos o más. Para recuperarse de un Region-wide error con un clúster secundario independiente,

es necesario crear una nueva instancia después de la conmutación por error. Una instancia nueva puede tardar entre 10 y 15 minutos en estar disponible.

Cómo añadir un clúster secundario independiente a tu clúster global

1. Inicie sesión en la consola de Amazon DocumentDB Consola de administración de AWS y ábrala.
2. En el panel de navegación izquierdo, elija Clústeres.
3. Elija el clúster global que necesita un clúster secundario. Asegúrese de que el clúster principal es.`Available`
4. En Acciones, elija Agregar región.
5. En la página Agregar una región, elige la región secundaria.

   > **Note**
   >
   > No puedes elegir una región que ya tenga un clúster secundario para el mismo clúster global. Además, no puede ser la misma región que el clúster principal.

6. Complete los campos restantes para el clúster secundario de en la nueva región de . Estas son las mismas opciones de configuración que para cualquier instancia de clúster.
7. Agrega una región. Cuando termines de agregar la región a tu clúster global, aparecerá en la lista de `Clusters` Consola de administración de AWS.
8. Compruebe el estado del clúster secundario y de su instancia lectora antes de continuar. Para ello, utilice el Consola de administración de AWS o el AWS CLI. A continuación, se muestra un ejemplo de comando si usa la AWS CLI:

   ```
   $ aws docdb describe-db-clusters --db-cluster-identifier secondary-cluster-id --
   query '*[].[Status]' --output text
   ```

   El estado de un clúster secundario recién agregado puede tardar varios minutos en cambiar de “creación en curso” a “disponible”. Cuando el clúster se encuentra disponible, puede eliminar la instancia de lector.
9. Seleccione la instancia de lector en el clúster secundario y, a continuación, elija Eliminar.
10. Después de eliminar la instancia de lector, el clúster secundario sigue siendo parte del clúster global. No debe tener ninguna instancia asociada con él.

> **Note**
>
> Puede utilizar este clúster secundario de Amazon DocumentDB independiente para recuperar manualmente su clúster global de Amazon DocumentDB de una interrupción no planificada en la región principal si se produce una interrupción de este tipo.

Conexión a un clúster global de Amazon DocumentDB

La forma de conectarse a un clúster global depende de si necesita escribir en el clúster o leer del clúster:

- Para solicitudes o consultas de solo lectura, debe conectarse al punto de conexión del lector para el clúster en Región de AWS.
- Para ejecutar instrucciones en lenguaje de manipulación de datos (DML) o lenguaje de definición de datos (DDL), conecte el punto de conexión del clúster para el clúster principal. Este punto final puede estar en una aplicación Región de AWS diferente a la suya.

Cuando visualice un clúster global en la consola, podrá ver todos los puntos de conexión de uso general asociados a todos sus clústeres.

La forma de conectarse a un clúster global depende de si necesita escribir en la base de datos o leer de ella. En el caso de las operaciones de DDL, DML y lectura que desee ejecutar desde la región principal, conéctese a su clúster principal utilizando el punto de enlace del clúster en modo conjunto de réplicas, con una preferencia de lectura de. `secondaryPreferred=true` Esto dirige el tráfico de escritura a la instancia de escritura del clúster principal y el tráfico de lectura a la instancia de réplica del clúster principal.

Para el tráfico interregional de solo lectura, conéctese a uno de sus clústeres secundarios mediante el punto de enlace del clúster en modo conjunto de réplicas. Como todas las instancias son réplicas de solo lectura, especifique una preferencia de lectura que no sea `primary` (por ejemplo,, `secondary` o). `secondaryPreferred` `nearest` Para minimizar la latencia, elija el punto de enlace del lector en su región de o en la región de que tenga más cerca.

Supervisión de clústeres globales de Amazon DocumentDB

Amazon DocumentDB (compatible con MongoDB) se integra CloudWatch para que pueda recopilar y analizar las métricas operativas de sus clústeres. Puede supervisar estas métricas mediante la

CloudWatch consola, la consola de Amazon DocumentDB, la AWS Command Line Interface (AWS CLI) o la API. CloudWatch

Para supervisar un clúster global, utilice las siguientes CloudWatch métricas.

Métrica	Description (Descripción)
`GlobalClusterReplicatedWriteIO`	El número promedio de I/O operaciones de escritura facturadas que se replican del volumen del clúster principal Región de AWS al volumen del clúster de un secundario Región de AWS, registrado en intervalos de 5 minutos. El número de réplicas en `ReplicatedWriteIOs` cada región secundaria es el mismo que el número de operaciones dentro de la región `VolumeWriteIOPs` realizadas por la región principal.
`GlobalClusterDataTransferBytes`	La cantidad de datos transferidos del clúster principal Región de AWS a un clúster secundario Región de AWS, medida en bytes.
`GlobalClusterReplicationLag`	La cantidad de retraso, en milisegundos, al replicar los eventos de cambio del clúster principal Región de AWS al de un clúster secundario Región de AWS

Para obtener más información sobre cómo ver estas métricas, consulta Visualización de CloudWatch datos.

Recuperación ante desastres y clústeres globales de Amazon DocumentDB

Temas

- Ejecución de una conmutación por error administrada para un clúster global de Amazon DocumentDB
- Ejecución de una conmutación por error manual para un clúster global de Amazon DocumentDB
- Ejecución de una transición para un clúster global de Amazon DocumentDB

- Desbloqueo de una conmutación por error o una transición de clúster global
- Administración de los RPO para los clústeres globales de Amazon DocumentDB

Al usar un clúster global, puedes recuperarte rápidamente de desastres, como los errores regionales. La recuperación de desastres suele medirse mediante valores para RTO y RPO.

- Objetivo de tiempo de recuperación (RTO) – El tiempo que tarda un sistema en volver a un estado operativo después de un desastre. En otras palabras, el RTO mide el tiempo de inactividad. Para un clúster global, RTO en minutos.
- Objetivo de punto de recuperación (RPO) – La cantidad de datos que se pueden perder (medidos en el tiempo). Para un clúster global, el RPO suele medirse en segundos.
- Para recuperarte de una interrupción no planificada, puedes realizar una conmutación por error entre regiones a una de las unidades secundarias de tu clúster global. Si tu clúster global tiene varias regiones secundarias, asegúrate de separar todas las regiones secundarias que quieras promover como principales. Luego, promocionas una de esas regiones secundarias para convertirla en la nueva región principal. Región de AWS Por último, crea nuevos clústeres en cada una de las demás regiones secundarias y vincula esos clústeres a tu clúster global.

Ejecución de una conmutación por error administrada para un clúster global de Amazon DocumentDB

Este enfoque tiene por objeto garantizar la continuidad empresarial en caso de que se produzca un verdadero desastre regional o una interrupción total del nivel de servicio.

Durante una conmutación por error administrada, el clúster principal se conmuta por error a la región secundaria que elija mientras se mantiene la topología de reproducción existente del clúster global de Amazon DocumentDB. El clúster secundario elegido promueve uno de sus nodos de solo de lectura al estado de escritor completo. Este paso permite que el clúster asuma el rol de clúster principal. La base de datos no estará disponible durante un breve periodo, mientras el clúster asume su nuevo rol. Los datos que no se replicaron del clúster principal anterior al clúster secundario elegido se pueden perder cuando este clúster secundario se convierta en el nuevo clúster principal. El volumen principal anterior hace todo lo posible por tomar una instantánea antes de sincronizarla con el nuevo volumen principal, de modo que los datos no replicados se conserven en la instantánea.

> **Note**
>
> Solo puede hacer una conmutación por error administrada del clúster entre regiones en un clúster global de Amazon DocumentDB si el clúster principal y todos los clústeres secundarios tienen las mismas versiones del motor. Si las versiones del motor no son compatibles, puede realizar la conmutación por error manualmente por medio de los pasos que se indican en Ejecución de una conmutación por error manual para un clúster global de Amazon DocumentDB.
> Si las versiones del motor de la región no coinciden, se bloqueará la conmutación por error. Comprueba si hay actualizaciones pendientes y aplícalas para asegurarte de que todas las versiones del motor de la región coinciden y de que la conmutación por error del clúster global está desbloqueada. Para obtener más información, consulte Desbloqueo de una conmutación por error o una transición de clúster global.

Para minimizar la pérdida de datos, haga lo siguiente antes de usar esta función:

- Desconecte las aplicaciones para evitar que se envíen escrituras al clúster principal del clúster global de Amazon DocumentDB.
- Compruebe los tiempos de retraso para todos los clústeres secundarios de Amazon DocumentDB. La elección de la región secundaria con el menor retraso de replicación puede minimizar la pérdida de datos con respecto a la región principal que actualmente presenta errores. Compruebe los tiempos de retraso de todos los clústeres secundarios de Amazon DocumentDB del clúster global consultando la `GlobalClusterReplicationLag` métrica en Amazon CloudWatch. Estas métricas muestran el retraso (en milisegundos) de la replicación a un clúster secundario con respecto al clúster principal.

  Para obtener más información sobre CloudWatch las métricas de Amazon DocumentDB, consulte. Métricas de Amazon DocumentDB

Durante una conmutación por error administrada, el clúster secundario elegido se promueve a su nuevo rol de clúster principal. Sin embargo, no hereda las diversas opciones de configuración del clúster principal. Una falta de coincidencia en la configuración puede provocar problemas de rendimiento, incompatibilidades de carga de trabajo y otros comportamientos anómalos. Para evitar estos problemas, resuelva las diferencias entre los clústeres globales de Amazon DocumentDB en relación con lo siguiente:

- Si es necesario, configure un grupo de parámetros de clúster de Amazon DocumentDB para el nuevo clúster principal. Puede configurar los grupos de parámetros del clúster de Amazon DocumentDB de forma independiente para cada clúster de su clúster global de Amazon DocumentDB. Por lo tanto, cuando se promueve un clúster secundario para que asuma el rol principal, su grupo de parámetros puede configurarse de manera diferente que para el principal. Si es así, modifique el grupo de parámetros del clúster secundario promocionado para que se ajuste a la configuración del clúster principal. Para aprender a hacerlo, consulte Modificación de grupos de parámetros de clúster de Amazon DocumentDB.
- Configure las herramientas y opciones de monitoreo, como CloudWatch los eventos y las alarmas de Amazon: configure el clúster promocionado con la misma capacidad de registro, alarmas, etc., según sea necesario para el clúster global. Al igual que con los grupos de parámetros, la configuración de estas características no se hereda del clúster principal durante el proceso de conmutación por error. Algunas CloudWatch métricas, como el retraso de replicación, solo están disponibles para las regiones secundarias. Por lo tanto, una conmutación por error cambia la forma de ver esas métricas y configurar las alarmas en ellas, y podría requerir cambios en los paneles predefinidos. Para obtener más información sobre los clústeres de Amazon DocumentDB y la supervisión, consulte Supervisión y registro en Amazon DocumentDB.

Por lo general, el clúster secundario elegido asume el rol principal en menos de un minuto. En cuanto el nodo de escritor de la nueva región principal esté disponible, podrá conectar sus aplicaciones a él y reanudar sus cargas de trabajo. Una vez que Amazon DocumentDB promueve el nuevo clúster principal, reconstruye automáticamente todos los clústeres regionales secundarios adicionales.

Como los clústeres globales de Amazon DocumentDB utilizan la replicación asíncrona, el retraso de la replicación en cada región secundaria puede variar. Amazon DocumentDB reconstruye estas regiones secundarias para que tengan exactamente los mismos datos de un momento dado que el nuevo clúster de región principal. La duración de la tarea de reconstrucción completa puede tardar entre unos minutos y varias horas, según el tamaño del volumen de almacenamiento y la distancia entre las regiones. Cuando los clústeres de la región secundaria terminen de reconstruirse a partir de la nueva región principal, estarán disponibles para el acceso de lectura. Tan pronto como se promocione y esté disponible el nuevo escritor principal, el clúster de la nueva región principal podrá gestionar las operaciones de lectura y escritura del clúster global de Amazon DocumentDB.

Para restaurar la topología original del clúster global, Amazon DocumentDB supervisa la disponibilidad de la antigua región principal. Tan pronto como la región esté en buen estado y vuelva a estar disponible, Amazon DocumentDB volverá a agregarla automáticamente al clúster global como región secundaria. Antes de crear el nuevo volumen de almacenamiento en la

antigua región principal, Amazon DocumentDB intenta tomar una instantánea del volumen de almacenamiento anterior en el punto en que se produjo el error. Lo hace para que pueda usarla para recuperar cualquiera de los datos perdidos. Si esta operación se realiza correctamente, Amazon DocumentDB coloca esta instantánea denominada «rds:docdb-unplanned-global-failovername-of-old-primary-» en la sección de instantáneas de. DB-cluster-timestamp Consola de administración de AWS También puede ver esta instantánea en la información devuelta por la operación de la API `DescribeDBClusterSnapshots`.

> **Note**
>
> La instantánea del volumen de almacenamiento anterior es una instantánea del sistema que está sujeta al período de retención de la copia de seguridad configurado en el clúster principal anterior. Para conservar esta instantánea más allá del período de retención, puede copiarla para guardarla como una instantánea manual. Para obtener más información sobre la copia de instantáneas, incluido el precio, consulte Copia de una instantánea de clúster.

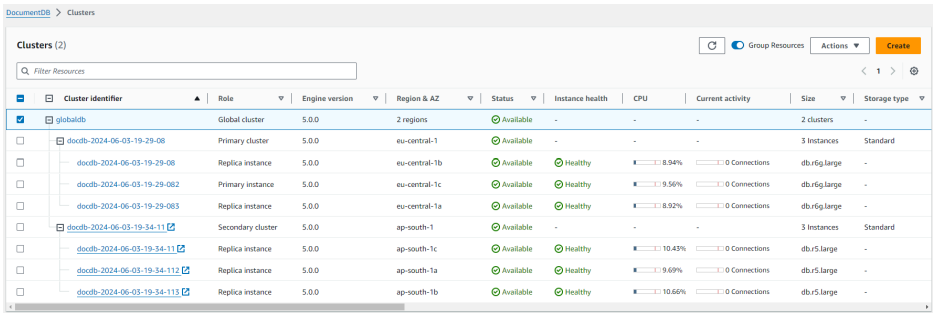
Una vez restaurada la topología original, puede conmutar por recuperación el clúster global a la región principal original mediante una operación de transición cuando sea más conveniente para su empresa y su carga de trabajo. Para ello, siga los pasos que se indican en Ejecución de una transición para un clúster global de Amazon DocumentDB.

Puede conmutar por error su clúster global de Amazon DocumentDB mediante la API, la o la API de Amazon DocumentDB. Consola de administración de AWS AWS CLI

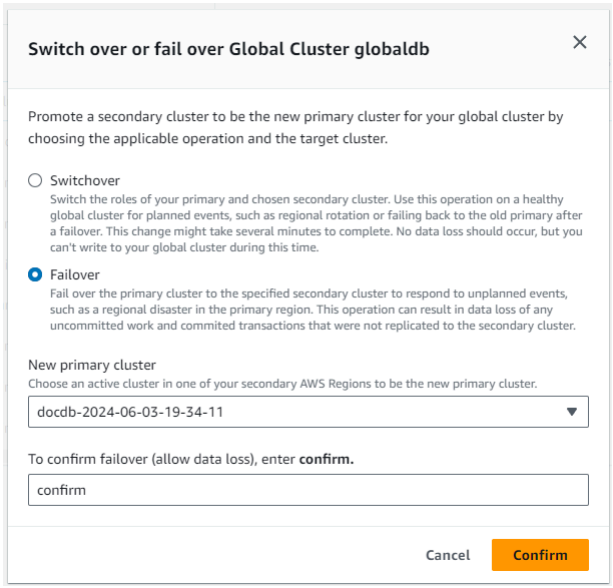
Using the Consola de administración de AWS

Ejecución de una conmutación por error administrada en el clúster global de Amazon DocumentDB

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Clusters (Clústeres).
3. Busque y elija el clúster global de Amazon DocumentDB que desea someter al proceso de conmutación por error.

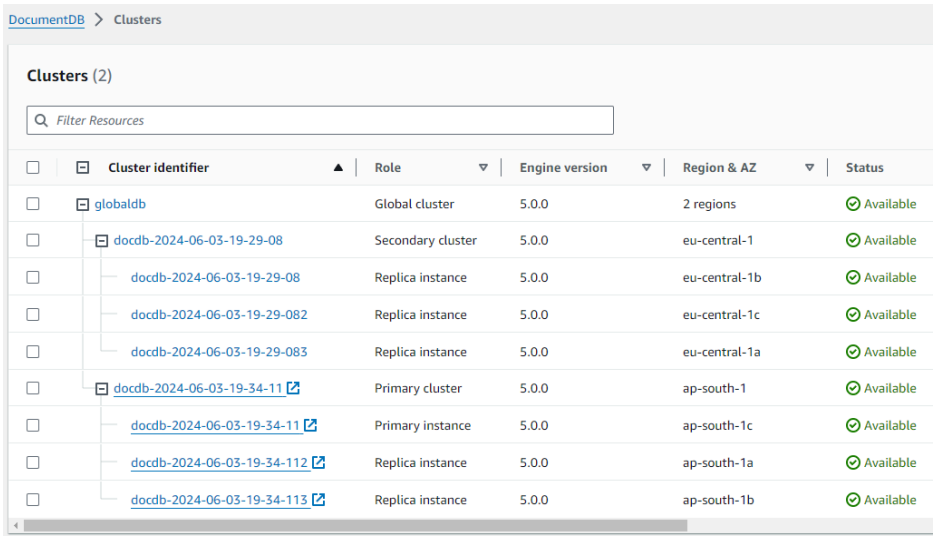


4. Elija Transición o conmutación por error en el menú Acciones.
5. En el cuadro de diálogo que aparece, seleccione Conmutación por error y, a continuación, elija el clúster secundario en la lista desplegable del campo Nuevo clúster principal.



6. Escriba “confirmar” en el último campo. A continuación, seleccione Confirm (Confirmar).

   El estado del clúster principal cambia a "Failing-over». Esta condición puede necesitar aproximadamente un minuto. Durante este tiempo, el estado del nuevo clúster principal muestra “Modificando…”. Una vez que se promocione el nuevo clúster principal, aparecerá como “Disponible” y podrá procesar transacciones de lectura y escritura. Las regiones secundarias, incluida la antigua principal, mostrarán «Resincronizando... «mientras se resincroniza con la nueva primaria. Al igual que el nuevo clúster principal, solo podrá procesar transacciones una vez que su estado cambie a “Disponible”.
7. Cuando se complete, el clúster principal original se convierte en el clúster secundario. El clúster secundario seleccionado se convierte en el clúster principal.



Using the AWS CLI

Ejecución de una conmutación por error administrada en el clúster global de Amazon DocumentDB

Ejecute el comando `failover-global-cluster` de la CLI para conmutar por error el clúster global de Amazon DocumentDB. Con el comando, pase valores para las siguientes opciones:

- `--region`
- `--global-cluster-identifier`
- `--target-db-cluster-identifier`
- `--allow-data-loss`

En los siguientes ejemplos, sustituya cada uno por la información *`user input placeholder`* de su clúster.

Para 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
```

Para 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
```

Ejecución de una conmutación por error manual para un clúster global de Amazon DocumentDB

Si un clúster completo de uno Región de AWS deja de estar disponible, puedes promocionar otro clúster del clúster global para que tenga read/write capacidad.

Puede activar manualmente el mecanismo de conmutación por error del clúster global si un clúster de una Región de AWS diferente es una mejor opción para ser el clúster primario. Por ejemplo, puede aumentar la capacidad de uno de esos clústeres secundarios y promoverlo para que sea el clúster principal. O bien, el equilibrio de actividad entre ellos Regiones de AWS podría cambiar, por lo que cambiar el clúster principal a otro Región de AWS podría reducir la latencia de las operaciones de escritura.

El siguiente procedimiento describe qué hacer para promocionar uno de los clústeres secundarios de un clúster global de Amazon DocumentDB.

Para promover un clúster secundario:

1. Deje de emitir sentencias DML y otras operaciones de escritura en el clúster principal durante Región de AWS la interrupción.
2. Identifique un clúster de un secundario Región de AWS para usarlo como un clúster principal nuevo. Si tienes dos (o más) secundarios Regiones de AWS en tu clúster global, elige el clúster secundario que tenga el menor retraso.
3. Desconecte el clúster secundario del clúster global elegido.

   Al eliminar un clúster secundario de un clúster global, se detiene inmediatamente la replicación del clúster principal a este secundario y lo convierte en un clúster aprovisionado independiente con todas read/write las capacidades. Cualquier otro clúster secundario asociado al clúster principal de la región donde se produjo la interrupción seguirá estando disponible y puede aceptar llamadas desde su aplicación. También consumen recursos. Dado que está recreando el clúster global, para evitar problemas de split-brain y otros problemas, elimine

los otros clústeres secundarios antes de crear el nuevo clúster en los pasos que se indican a continuación.

Para obtener más información sobre los pasos para desasociar clústeres, consulte Eliminación de un clúster global de Amazon DocumentDB.

4. Este clúster se convierte en el clúster primario de un nuevo clúster global cuando comienza a agregarle regiones, en el siguiente paso.
5. Agregue un Región de AWS al clúster. Al hacerlo, comienza el proceso de reproducción de clúster principal a secundario.
6. Agregue más Regiones de AWS según sea necesario para volver a crear la topología necesaria para respaldar su aplicación. Asegúrese de que las escrituras de la aplicación se envían al clúster correcto antes, durante y después de realizar cambios como estos, para evitar incoherencias de datos entre los clústeres en el clúster global (problemas de split-brain).
7. Cuando se haya resuelto la interrupción y esté listo para asignar su Región de AWS original como clúster primario de nuevo, realice los mismos pasos en orden inverso.
8. Elimine uno de los clústeres secundarios del clúster global. Esto le permitirá atender read/write el tráfico.
9. Redirija todo el tráfico de escritura del clúster primario en la Región de AWS original.
10. Agregue un Región de AWS para configurar uno o más clústeres secundarios de la Región de AWS misma manera que antes.

Los clústeres globales de Amazon DocumentDB se pueden administrar mediante AWS SDK, lo que le permite crear soluciones para automatizar el proceso de conmutación por error global de los clústeres en casos de uso de la recuperación ante desastres y la planificación de la continuidad empresarial. Una de estas soluciones está disponible para nuestros clientes con las licencias de Apache 2.0 y se puede acceder a ella desde nuestro repositorio de herramientas aquí. Esta solución aprovecha Amazon Route 53 para la administración de puntos finales y proporciona AWS Lambda funciones que se pueden activar en función de los eventos apropiados.

Ejecución de una transición para un clúster global de Amazon DocumentDB

Al utilizar las transiciones, puede cambiar la región del clúster principal de forma rutinaria. Este enfoque está destinado a situaciones controladas, como el mantenimiento operativo y otros procedimientos operativos planificados.

Existen tres casos de uso frecuentes en los que se utilizan las transiciones:

- Para los requisitos de “rotación regional” impuestos a sectores específicos. Por ejemplo, es posible que los reglamentos de los servicios financieros exijan que los sistemas de nivel 0 se cambien a una región diferente durante varios meses para garantizar que los procedimientos de recuperación ante desastres se ensayen con cierta asiduidad.
- Para aplicaciones multirregionales del tipo “seguir el sol”. Por ejemplo, es posible que una empresa desee ofrecer escrituras con menor latencia en diferentes regiones en función del horario laboral en distintas zonas horarias.
- Como método sin pérdida de datos para conmutar por recuperación a la región principal original tras una conmutación por error.

> **Note**
>
> Las transiciones están diseñadas para utilizarse en un clúster global de Amazon DocumentDB en buen estado. Para la recuperación de una interrupción no programada, siga el procedimiento correspondiente en Ejecución de una conmutación por error manual para un clúster global de Amazon DocumentDB.
> Para realizar un cambio, todas las regiones secundarias deben ejecutar exactamente la misma versión del motor que la principal. Si las versiones del motor de la región no coinciden, se bloqueará el cambio. Comprueba si hay actualizaciones pendientes y aplícalas para asegurarte de que todas las versiones del motor de la región coinciden y de que el cambio de clúster global está desbloqueado. Para obtener más información, consulte Desbloqueo de una conmutación por error o una transición de clúster global.

Durante una transición, Amazon DocumentDB cambia el clúster principal a la región secundaria que elija a la vez que mantiene la topología de replicación existente del clúster global. Antes de iniciar el proceso de transición, Amazon DocumentDB espera a que todos los clústeres de regiones secundarias estén completamente sincronizados con el clúster de la región principal. A continuación, el clúster de bases de datos de la región principal se convierte en un clúster de solo lectura y el clúster secundario que elija promueve uno de sus nodos de solo lectura a estado de escritor completo. Al convertir este nodo en escritor, el clúster secundario puede asumir el rol de clúster principal. Dado que todos los clústeres secundarios se sincronizaron con el principal al principio del proceso, el nuevo principal continúa las operaciones para el clúster global de Amazon DocumentDB sin perder ningún dato. La base de datos no estará disponible durante un breve periodo, mientras los clústeres principales y secundarios seleccionados asumen nuevas funciones.

Para optimizar la disponibilidad de las aplicaciones, haga lo siguiente antes de usar esta función:

- Lleve a cabo esta operación durante los horarios menos concurridos o en otro momento cuando las escrituras en el clúster principal sean mínimas.
- Desconecte las aplicaciones para evitar que se envíen escrituras al clúster principal del clúster global de Amazon DocumentDB.
- Compruebe los tiempos de retraso de todos los clústeres secundarios de Amazon DocumentDB del clúster global consultando la `GlobalClusterReplicationLag` métrica en Amazon CloudWatch. Esta métrica muestra el retraso (en milisegundos) de la replicación a un clúster secundario con respecto al clúster principal. Este valor es directamente proporcional al tiempo que tarda Amazon DocumentDB en completar la transición. Por lo tanto, cuanto mayor sea el valor de retraso, más tiempo llevará la transición.

  Para obtener más información sobre CloudWatch las métricas de Amazon DocumentDB, consulte. Métricas de Amazon DocumentDB

Durante una transición, el clúster secundario de base de datos elegido se promueve a su nuevo rol de clúster principal. Sin embargo, no hereda las diversas opciones de configuración del clúster principal de base de datos. Una falta de coincidencia en la configuración puede provocar problemas de rendimiento, incompatibilidades de carga de trabajo y otros comportamientos anómalos. Para evitar estos problemas, resuelva las diferencias entre los clústeres globales de Amazon DocumentDB en relación con lo siguiente:

- Configure un grupo de parámetros de clúster de base de datos de Amazon DocumentDB para el nuevo clúster principal, si es necesario: puede configurar los grupos de parámetros de clúster de Amazon DocumentDB de forma independiente para cada clúster del clúster global de Amazon DocumentDB. Esto significa que cuando se promueve un clúster secundario de base de datos para asumir el rol principal, su grupo de parámetros puede configurarse de manera diferente que para el principal. Si es así, modifique el grupo de parámetros del clúster secundario de base de datos promocionado para que se ajuste a la configuración del clúster principal. Para saber cómo hacerlo, consulte Administración de los grupos de parámetros de clúster de Amazon DocumentDB.
- Configure las herramientas y opciones de monitoreo, como CloudWatch los eventos y las alarmas de Amazon: configure el clúster promocionado con la misma capacidad de registro, alarmas, etc., según sea necesario para el clúster global. Al igual que con los grupos de parámetros, la configuración de estas características no se hereda del clúster principal durante el proceso de transición. Algunas CloudWatch métricas, como el retraso de replicación, solo están disponibles para las regiones principales. Por lo tanto, una transición cambia la forma de ver esas métricas

y configurar las alarmas en ellas, y podría requerir cambios en los paneles predefinidos. Para obtener más información, consulte Supervisión y registro en Amazon DocumentDB.

> **Note**
>
> Por lo general, la transición de rol puede tardar varios minutos.

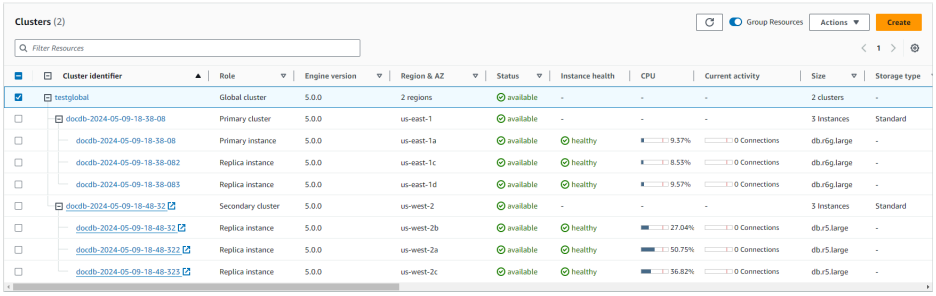
Cuando finaliza el proceso de transición, el clúster de Amazon DocumentDB promocionado puede manejar operaciones de escritura para el clúster global.

Puede cambiar su clúster global de Amazon DocumentDB mediante las siguientes Consola de administración de AWS opciones: AWS CLI

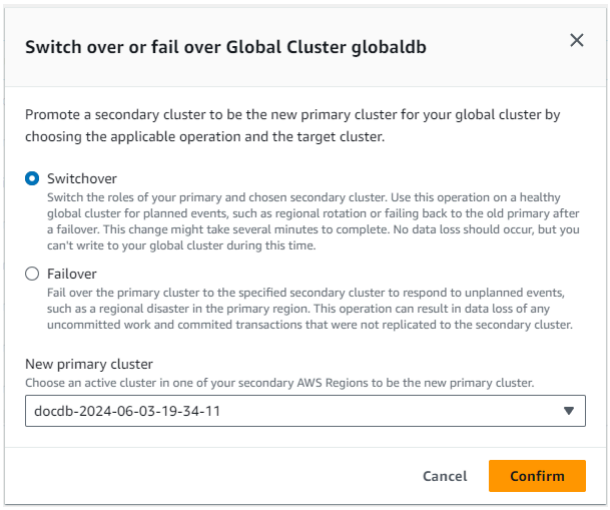
Using the Consola de administración de AWS

Ejecución de una transición en un clúster global de Amazon DocumentDB

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Clusters (Clústeres).
3. Busque y elija el clúster global de Amazon DocumentDB que desea someter al proceso de transición.



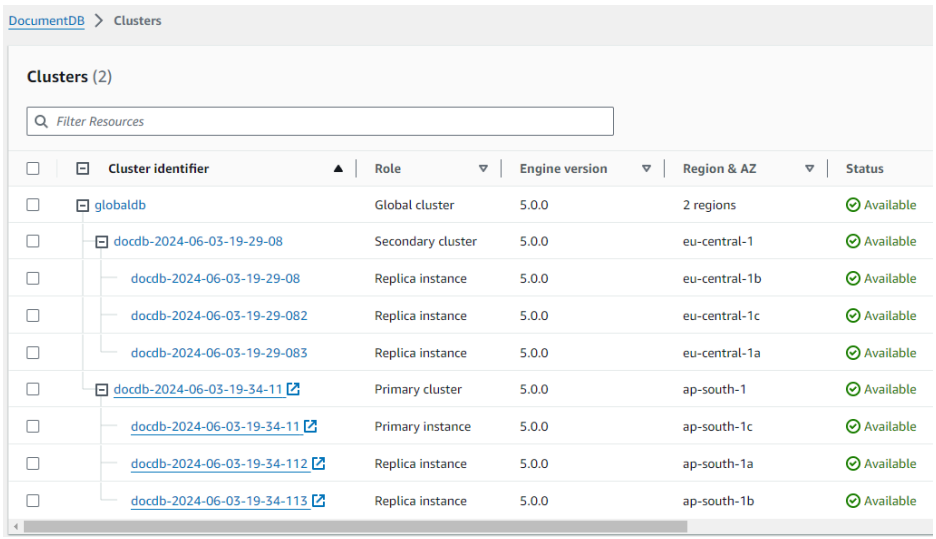
4. Elija Transición o conmutación por error en el menú Acciones.
5. En el cuadro de diálogo que aparece, seleccione Transición y, a continuación, elija el clúster secundario en la lista desplegable del campo Nuevo clúster principal.



6. Elija Confirmar.

   El estado del clúster principal cambia a "Switching-over». Esta condición puede necesitar aproximadamente tres minutos. Durante este tiempo, el estado de todos los clústeres regionales muestra “Modificando...”. Una vez que las regiones estén sincronizadas y se promueva el nuevo servidor principal, aparecerá la palabra «Disponible» en todos los campos de estado y podrá atender las transacciones.

7. Cuando se complete, el clúster principal original se convierte en el clúster secundario. El clúster secundario seleccionado se convierte en el clúster principal.



Using the AWS CLI

Ejecución de una transición en un clúster global de Amazon DocumentDB

Utilice el comando `switchover-global-cluster` de la CLI para hacer la transición del clúster global de Amazon DocumentDB. Con el comando, pase valores para las siguientes opciones:

- `--region`
- `--global-cluster-identifier`
- `--target-db-cluster-identifier`

En los siguientes ejemplos, sustituya cada uno por *`user input placeholder`* la información de su clúster.

Para 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
```

Para 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
```

Desbloqueo de una conmutación por error o una transición de clúster global

Regiones en diferentes versiones del motor

Las transiciones y las conmutaciones por error de clústeres globales se bloquean cuando no todos los clústeres regionales del clúster global están en la misma versión del motor. Si las versiones no coinciden, es posible que aparezca este error al realizar una conmutación o una conmutación por error: el clúster de base de datos de destino especificado ejecuta una versión de motor con un nivel de parche diferente al del clúster de base de datos de origen. Aplica de forma rutinaria las versiones más recientes del motor para mantener los clústeres globales en buen estado.

Para resolver este error, actualice primero todas las regiones secundarias y, después, la región principal a la misma versión del motor. Para ello, aplique las acciones de mantenimiento pendientes.

Para ver las acciones de mantenimiento pendientes y aplicar los cambios necesarios para corregir el problema, siga las instrucciones de una de las siguientes pestañas:

Using the Consola de administración de AWS

Para desbloquear una transición o conmutación por error de clúster global, debe determinar si hay acciones de mantenimiento pendientes para sus clústeres y aplicarlas. Siga estos pasos para ver y aplicar las acciones de mantenimiento:

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Clusters (Clústeres).
3. En la tabla Clústeres, localice el clúster global en la columna de Identificador de clúster. En su clúster global, anote cada clúster secundario y el clúster principal del clúster global en cuestión, y realice los siguientes pasos para cada uno.
4. Para cada clúster secundario:
    a. Si hay una actualización disponible para el clúster, se indicará con la palabra Disponible, Obligatorio o Siguiente período en la columna Mantenimiento.
    b. Para realizar una acción, elija el clúster para mostrar sus detalles y, a continuación, seleccione Mantenimiento y copias de seguridad. Aparecerán los elementos de mantenimiento pendientes.
    c. En Descripción, si indica que hay una «Nueva actualización de mantenimiento disponible», selecciónela y, a continuación, elija Aplicar ahora.
5. Para su clúster principal:
    a. Si hay una actualización disponible para el clúster, se indicará con la palabra Disponible, Obligatorio o Siguiente período en la columna Mantenimiento.
    b. Para realizar una acción, elija el clúster para mostrar sus detalles y, a continuación, seleccione Mantenimiento y copias de seguridad. Aparecerán los elementos de mantenimiento pendientes.
    c. En Descripción, si indica que hay una «Nueva actualización de mantenimiento disponible», selecciónela y, a continuación, elija Aplicar ahora.

Using the AWS CLI

Para desbloquear una transición o conmutación por error de clúster global, debe determinar si hay acciones de mantenimiento pendientes para el clúster y aplicarlas. Siga estos pasos para ver y aplicar las acciones de mantenimiento primero en los clústeres secundarios y, después, en el clúster principal del clúster global:

1. Ejecute primero lo siguiente en el clúster regional de cada región secundaria y, a continuación, en el clúster regional principal de las regiones.
2. Ejecute el comando `describe-pending-maintenance-actions` de la CLI con la opción `--resource-identifier` para determinar si hay acciones de mantenimiento disponibles para su clúster regional de Amazon DocumentDB.

   En los siguientes ejemplos, reemplaza cada uno *`user input placeholder`* por la información de tu clúster.

   Para 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
```

   Para Windows:

```
aws docdb describe-pending-maintenance-action ^
   --resource-identifier arn:aws:rds:us-
east-1:001234567890:cluster:docdb-2025-03-27-19-21-15
```

   El resultado tendrá un aspecto parecido al siguiente:

```
{
    "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. Si es necesaria una acción de mantenimiento, ejecute el comando [apply-pending-maintenance-action] de la CLI con las siguientes opciones:

   - `--resource-identifier`
   - `--apply-action`
   - `--opt-in-type`
   - `--region`

   En los siguientes ejemplos, sustituya cada uno *`user input placeholder`* por la información de su clúster.

   Para 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
```

   Para 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 vez finalizada la acción de mantenimiento, vuelva a ejecutar el comando `describe-pending-maintenance-actions` para asegurarse de que no haya otras acciones pendientes para el clúster.

   El resultado que desea es:

```
{
    "PendingMaintenanceActions": []
}
```

Using the Amazon DocumentDB API

Para desbloquear una transición o conmutación por error de clúster global, debe determinar si hay acciones de mantenimiento pendientes para el clúster y aplicarlas. Use las siguientes API para ver y aplicar las acciones de mantenimiento:

1. Ejecuta primero lo siguiente en el clúster regional de cada región secundaria y, después, en el clúster regional principal de las regiones.
2. Llame a la API de PendingMaintenanceAction para determinar si hay acciones de mantenimiento disponibles para su clúster global de Amazon DocumentDB.
3. Para aplicar cualquier cambio, llame a la API de ApplyPendingMaintenanceAction.

Cambios pendientes en el clúster de destino

Las conmutaciones y las conmutaciones por error también se bloquean cuando el clúster secundario de destino tiene un cambio programado que aún no se ha aplicado, como una acción de modificación o mantenimiento que hayas solicitado para el siguiente período de mantenimiento. En este caso, es posible que aparezca este error al realizar una conmutación o una conmutación por error: no puedes realizar una conmutación por error al clúster con ARN *arn:aws:rds:us-east-1:001234567890:cluster:docdb-2025-03-27-19-21-15* porque se está modificando o porque hay modificaciones pendientes. Vuelva a intentarlo cuando el clúster esté disponible. Para desbloquear la operación, elimine el cambio programado en el clúster de destino. Los pasos dependen del tipo de cambio. Cuando se elimine el cambio programado y el estado del clúster vuelva a ser`available`, vuelva a intentar la conmutación o la conmutación por error.

Cancela una acción de mantenimiento programada

No puedes cancelar una acción de mantenimiento programada (deshacer una suscripción) desde Consola de administración de AWS, así que usa el AWS CLI.

Using the AWS CLI

1. Ejecute el comando describe-pending-maintenance-actions de la CLI con la `--resource-identifier` opción y busque una acción de mantenimiento que sea. `OptInStatus` `next-maintenance`

   En los siguientes ejemplos, sustituya cada uno por la información de su clúster. *`user input placeholder`*

   Para 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
   ```

   Para 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
   ```

   El resultado es similar al siguiente. Anote el `Action` valor (`os-upgrade`en este ejemplo) que utilizará en el paso siguiente.

   ```
   {
       "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. Para cancelar la acción programada, ejecute el comando apply-pending-maintenance-action de la CLI y transfiera el valor del paso `--opt-in-type undo-opt-in` anterior a. `Action` `--apply-action`

   Para 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
```

   Para 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
```

   En la respuesta, la acción de mantenimiento ya no tiene un, lo que confirma que se canceló la suscripción `OptInStatus` programada.

```
{
    "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"
            }
```

```
        ]
    }
}
```

Revertir una modificación de clúster programada

Using the AWS CLI

1. Compruebe si hay modificaciones programadas ejecutando el comando describe-db-clusters de la CLI e inspeccionando el campo de salida. `PendingModifiedValues`

   > **Note**
   >
   > Utilice la CLI de Amazon RDS (`aws rds`) para este comando, no la CLI de Amazon DocumentDB (`aws docdb`), porque la API de Amazon DocumentDB no devuelve ningún campo a nivel de clúster. `describe-db-clusters PendingModifiedValues`

   En los siguientes ejemplos, sustituya cada uno *`user input placeholder`* por la información de su clúster.

   Para 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
   ```

   Para Windows:

   ```
   aws rds describe-db-clusters ^
      --db-cluster-identifier docdb-2025-03-27-19-21-15 ^
      --query "DBClusters[0].PendingModifiedValues" ^
      --region us-east-1
   ```

El resultado muestra los cambios programados. En este ejemplo, está pendiente un cambio en el nombre del clúster (`DBClusterIdentifier`), de `docdb-2025-03-27-19-21-15` a`docdb-new-name`.

```
{
    "DBClusterIdentifier": "docdb-new-name"
}
```

2. Para revertir el cambio, ejecute el comando de la CLI modify-db-cluster con `--apply-immediately` y restablezca el valor modificado en su valor original. En este ejemplo, se revierte el cambio de nombre pendiente `--new-db-cluster-identifier` restableciendo el nombre actual del clúster.

   Para 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
```

   Para 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
```

Administración de los RPO para los clústeres globales de Amazon DocumentDB

Con un clúster global de Amazon DocumentDB, puede administrar el objetivo del punto de recuperación (RPO) mediante el parámetro. `global_db_rpo` El RPO representa la cantidad máxima de datos que se pueden perder en caso de interrupción.

Cuando establece un RPO para su clúster global de Amazon DocumentDB, Amazon DocumentDB monitoriza el tiempo de retraso del RPO de todos los clústeres secundarios. Esta supervisión garantiza que al menos un clúster secundario permanezca dentro del período de RPO objetivo.

La configuración del RPO controla la forma en que Amazon DocumentDB administra las transacciones de escritura en el clúster principal para limitar la posible pérdida de datos en caso de que se produzca una conmutación por error. Amazon DocumentDB evalúa los tiempos de retraso del RPO y del RPO para confirmar (o bloquear) las transacciones en el servidor principal de la siguiente manera:

- Confirma la transacción si al menos un clúster secundario de base de datos tiene un tiempo de retraso de RPO menor que el RPO.
- Bloquea la transacción si todos los clústeres secundarios de base de datos tienen tiempos de retraso de RPO mayores que el RPO.

En otras palabras, si todos los clústeres secundarios están por debajo del RPO objetivo, Amazon DocumentDB detiene las transacciones en el clúster principal. Amazon DocumentDB reanuda y confirma las transacciones pausadas tan pronto como el retraso de al menos un clúster de base de datos secundario cae por debajo del RPO. El resultado es que ninguna transacción se puede confirmar hasta que se cumpla el RPO.

El parámetro `global_db_rpo` es dinámico. Si decide que no quiere que todas las transacciones de escritura se detengan hasta que el retraso disminuya lo suficiente, puede restablecerlo rápidamente. En este caso, Amazon DocumentDB aplica el cambio tras un breve retraso.

> **Important**
>
> En una base de datos global con solo dos AWS regiones, recomendamos mantener el valor predeterminado del `global_db_rpo` parámetro en el grupo de parámetros de la región secundaria. De lo contrario, realizar una conmutación por error debido a la pérdida de la AWS región principal podría provocar que Amazon DocumentDB pausara las transacciones. En su lugar, espere a que Amazon DocumentDB termine de reconstruir el clúster en la antigua AWS región donde se produjo el error antes de cambiar este parámetro para imponer un RPO máximo.

Temas

- Establecimiento del objetivo de punto de recuperación
- Visualización del objetivo de punto de recuperación
- Desactivación del objetivo de punto de recuperación

Establecimiento del objetivo de punto de recuperación

El `global_db_rpo` parámetro controla la configuración del RPO de una base de datos de Amazon DocumentDB. Los valores válidos oscilan entre 20 segundos y 2.147.483.647 segundos (68 años). Elija un valor realista para satisfacer las necesidades de su empresa. Por ejemplo, es posible que desee permitir hasta 10 minutos para su RPO, en cuyo caso establece el valor en 600.

Puede establecer este valor para su clúster global de Amazon DocumentDB mediante la API Consola de administración de AWS, la API o la AWS CLI API de Amazon DocumentDB.

Using the Consola de administración de AWS

Para establecer el RPO

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. Elija el clúster principal de su clúster global de Amazon DocumentDB y abra la pestaña Configuración para buscar el grupo de parámetros del clúster de base de datos correspondiente.

   Los grupos de parámetros no se pueden editar directamente. En su lugar, puede hacer lo siguiente:

   - Cree un grupo de parámetros de clúster de base de datos personalizado con el grupo de parámetros predeterminado apropiado como punto de partida.
   - En su grupo de parámetros de clúster de base de datos personalizado, defina el valor del parámetro global_db_rpo para adaptarlo a su caso de uso. Los valores válidos van desde 20 segundos hasta el valor entero máximo de 2 147 483 647 (68 años).
   - Aplique el grupo de parámetros del clúster de base de datos modificado a su clúster de base de datos de Amazon DocumentDB.

Para obtener más información sobre la modificación de los grupos de parámetros del clúster de base de datos, consulteModificación de grupos de parámetros de clúster de Amazon DocumentDB.

Using the AWS CLI

Para establecer el parámetro `global_db_rpo`, utilice el comando de CLI modify-db-cluster-parameter-group. En el comando, especifique el nombre del grupo de parámetros del clúster principal y los valores del parámetro RPO.

En el ejemplo siguiente se establece el RPO en 600 segundos (10 minutos) para el grupo de parámetros de clúster principal de base de datos denominado `my_custom_global_parameter_group`.

Para 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"
```

Para 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

Para modificar el `global_db_rpo` parámetro, usa la operación de la ModifyDBClusterParameterGroup API.

Visualización del objetivo de punto de recuperación

El objetivo de punto de recuperación (RPO) de un clúster global se almacena en el `global_db_rpo` parámetro de cada clúster de base de datos.

Puede usar la CLI para ver el `global_db_rpo` parámetro de un clúster de base de datos de Amazon DocumentDB. Utilice la `--query` opción para devolver solo el `global_db_rpo` parámetro del grupo de parámetros.

Para 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']"
```

Para Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name my_custom_global_parameter_group ^
    --query "Parameters[?ParameterName=='global_db_rpo']"
```

El comando devuelve un resultado similar al siguiente.

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

Para obtener más información sobre la visualización de los parámetros del grupo de parámetros del clúster, consulteAdministración de los grupos de parámetros de clúster de Amazon DocumentDB.

Desactivación del objetivo de punto de recuperación

Para desactivar el RPO, restablezca el parámetro `global_db_rpo`. Puede restablecer los parámetros mediante la API Consola de administración de AWS AWS CLI, la API o la API de Amazon DocumentDB.

Using the Consola de administración de AWS

Para deshabilitar el RPO

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb

2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).
3. En la lista, elija el grupo de parámetros de clúster de base de datos principal.
4. Elija el botón de opción situado junto al parámetro global_db_rpo.
5. Seleccione Restablecer los valores predeterminados y confírmelo.

Para obtener más información sobre cómo restablecer un parámetro con la consola, consulteModificación de grupos de parámetros de clúster de Amazon DocumentDB.

Using the AWS CLI

Para restablecer el parámetro `global_db_rpo`, utilice el comando reset-db-cluster-parameter-group.

Para 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"
```

Para 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

Para restablecer el `global_db_rpo` parámetro, utilice la operación de la ResetDBClusterParameterGroup API.

Administración de clústeres de Amazon DocumentDB

Para administrar un clúster de Amazon DocumentDB, debe disponer de una política de IAM con los permisos del plano de control de Amazon DocumentDB. Estos permisos le permiten crear, modificar y eliminar clústeres e instancias. La política `AmazonDocDBFullAccess` proporciona todos los permisos necesarios para administrar un clúster de Amazon DocumentDB.

Los siguientes temas muestran cómo realizar diversas tareas cuando se trabaja con clústeres de Amazon DocumentDB, incluida la creación, eliminación, modificación, conexión y consulta de clústeres.

Temas

- Descripción de clúster
- Configuración del clúster de Amazon DocumentDB
- Configuraciones de almacenamiento en clústeres de Amazon DocumentDB
- Determinación del estado de un clúster
- Ciclo de vida del clúster de Amazon DocumentDB
- Escalado de clústeres de Amazon DocumentDB
- Clonación de un volumen de clúster de base de datos de Amazon DocumentDB
- Información general de la tolerancia a errores de clúster de Amazon DocumentDB

Descripción de clúster

Amazon DocumentDB divide la capacidad informática y de almacenamiento, y descarga la replicación de datos y las copias de seguridad en el volumen del clúster. Un volumen de clúster proporciona una capa de almacenamiento duradera, fiable y altamente disponible que replica los datos de seis formas distintas entre tres zonas de disponibilidad. Las réplicas permiten una mayor disponibilidad de los datos y del escalado de lectura. Cada clúster se puede escalar verticalmente hasta 15 réplicas.

Nombre	Description (Descripción)	Operaciones de API (Verbos)

Clúster	Se compone de una o varias instancias y de un volumen de almacenamiento del clúster que administra los datos para esas instancias.	`create-db-cluster`
		`delete-db-cluster`
		`describe-db-clusters`
		`modify-db-cluster`

Instancia	La lectura y escritura de los datos en el volumen de almacenamiento del clúster	`create-db-instance` `delete-db-instance`

Nombre	Description (Descripción)	Operaciones de API (Verbos)

	se realiza a través de las instancias. En cualquier clúster, hay dos tipos de instancias: principal y de réplica. Un clúster siempre tiene una instancia principal y puede tener 0-15 réplicas.	`describe-db-instances`
		`modify-db-instance`
		`describe-orderable-db-instance-options`
		`reboot-db-instance`

Volumen del clúster	Un volumen de almacenamiento de base de datos virtual que abarca tres zonas de disponibilidad, de modo que cada una de estas zonas tenga dos copias de los datos del clúster.	N/A
Instancia principal	Admite operaciones de lectura y escritura, y realiza todas las modificaciones de los datos en el volumen del clúster. Cada clúster tiene una instancia principal.	N/A

Nombre	Description (Descripción)	Operaciones de API (Verbos)
Instancia de réplica	Solo admite operaciones de lectura. Cada clúster de Amazon DocumentDB puede tener hasta 15 instancias de réplica, además de la instancia principal. Varias réplicas distribuyen la carga de trabajo real. Al colocar las réplicas en distintas zonas de disponibilidad, también puede aumentar la disponibilidad de la base de datos.	N/A
Punto de conexión de clúster	Un punto de conexión de un clúster de Amazon DocumentDB que se conecta a la instancia principal actual del clúster. Cada clúster de Amazon DocumentDB tiene un punto de conexión de clúster y una instancia principal.	N/A

Nombre	Description (Descripción)	Operaciones de API (Verbos)
Punto de conexión del lector	Un punto de conexión de un clúster de Amazon DocumentDB que se conecta a una de las réplicas disponibles de ese clúster. Cada clúster de base de datos de Amazon DocumentDB tiene un punto de conexión del lector. Si hay más de una réplica, el punto de conexión del lector dirige cada solicitud de conexión a una de las réplicas de Amazon DocumentDB.	N/A
Punto de conexión de instancia	Un punto de conexión de una instancia de un clúster de Amazon DocumentDB que se conecta a una instancia específica. Cada instancia de un clúster de base de datos, independientemente del tipo de instancia, tiene su propio punto de conexión de instancia único.	N/A

Configuración del clúster de Amazon DocumentDB

Al crear o modificar un clúster, es importante saber qué parámetros son inmutables y cuáles se pueden modificar una vez creado el clúster. La siguiente tabla muestra todas las configuraciones o parámetros que son específicos en un clúster. Tal y como se indica en la tabla; algunos de ellos se pueden modificar y otros no.

> **Note**
>
> Esta configuración no debe confundirse con los grupos de parámetros de clúster de Amazon DocumentDB y sus parámetros. Para obtener más información acerca de los grupos de parámetros de clúster, consulte Administración de los grupos de parámetros de clúster de Amazon DocumentDB.

Parámetro	Modificable	Notas
DBClusterIdentifier	Sí	Restricciones en la nomenclatura: • Debe tener [1-63] letras, números o guiones. • El primer carácter debe ser una letra. • No puede terminar con un guion ni contener dos guiones consecutivos. • Debe ser única para todos los clústeres de Amazon RDS, Amazon Neptune y Amazon DocumentDB por región. Cuenta de AWS
Engine	No	Debe ser `docdb`.
BackupRetentionPeriod	Sí	Debe estar comprendido entre [1-35] días.
DBClusterParameterGroupName	Sí	Restricciones en la nomenclatura: • La longitud es de [1-255] caracteres alfanuméricos. • El primer carácter debe ser una letra. • No puede terminar con un guion ni contener dos guiones consecutivos.
DBSubnetGroupName	No	Una vez creado un clúster, no puede modificar la subred del clúster.

Parámetro	Modificable	Notas
EngineVersion	No	El valor puede ser `5.0.0` (predeterminado), `4.0.0` o `3.6.0`.
KmsKeyId	No	Si decide cifrar el clúster, no puede cambiar la AWS KMS clave que utilizó para cifrar el clúster.
MasterUsername	No	Una vez creado un clúster, no puede modificar el `MasterUsername` . Restricciones en la nomenclatura: • La longitud es de [1-63] caracteres alfanumér icos. • El primer carácter debe ser una letra. • No puede ser una palabra reservada por el motor de base de datos.

MasterUserPassword	Sí	Restricciones:
		
		• La longitud es de 8 a 100 caracteres ASCII imprimibles.
		• Se puede utilizar cualquier carácter ASCII imprimible, excepto los siguientes:
		  • **/** (barra inclinada)
		  • **"** (comillas dobles)
		  • **@** (símbolo de arroba)

Port	Sí	El número de puerto se aplica a todas las instancias del clúster.
PreferredBackupWindow	Sí	
PreferredMaintenanceWindow	Sí	

Parámetro	Modificable	Notas
`StorageEncrypted`	No	Si elige cifrar el clúster, no se puede descifrar.
`StorageType`	Sí	El tipo de almacenamiento del clúster de base de datos: Estándar (`standard`) o I/O-Optim ized (`iopt1`). Valor predeterminado: `standard` Este parámetro se puede configurar con `CreateDBCluster` y. `ModifyDBCluster` Para obtener más información, consulte Configuraciones de almacenamiento en clústeres de Amazon DocumentDB.
`Tags`	Sí	
`VpcSecurityGroupIds`	No	Una vez creado un clúster, no puede modificar la VPC en la que reside el clúster.

Configuraciones de almacenamiento en clústeres de Amazon DocumentDB

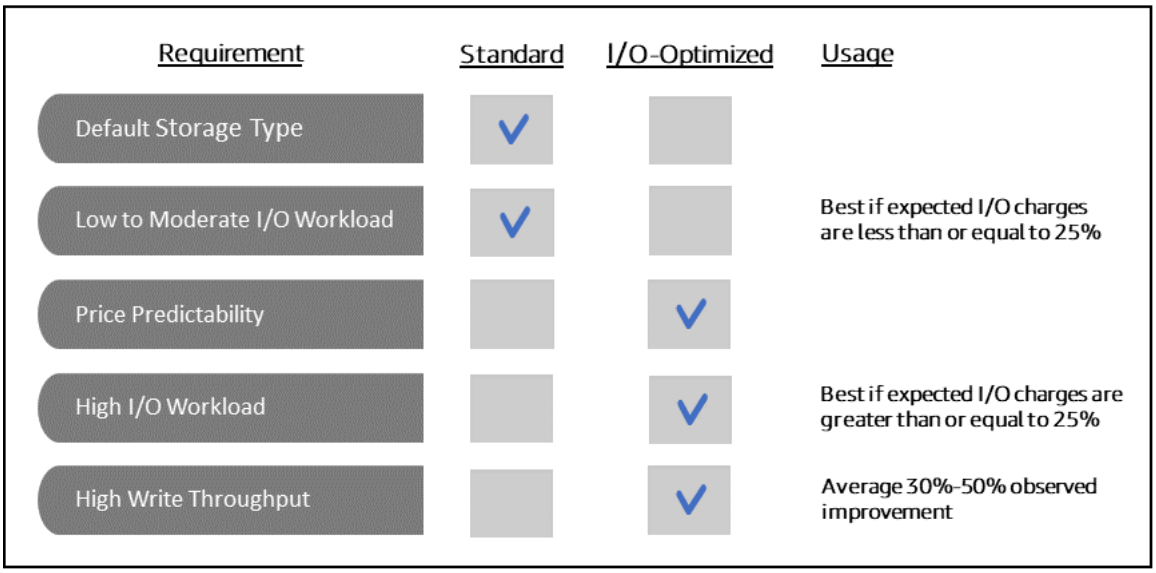
A partir de Amazon DocumentDB 5.0, los clústeres basados en instancias admiten dos tipos de configuraciones de almacenamiento:

- Almacenamiento estándar de Amazon DocumentDB: diseñado para clientes con un I/O consumo bajo o moderado. Si espera que sus I/O costos sean inferiores al 25% del total de su clúster de Amazon DocumentDB, esta opción podría ser ideal para usted. Con la configuración de almacenamiento estándar de Amazon DocumentDB, se le factura de I/O forma de pago por solicitud, además de los cargos de instancia y almacenamiento. Esto significa que su facturación puede variar de un ciclo a otro en función del uso. La configuración está diseñada para adaptarse a las cambiantes demandas de su aplicación I/O .
- I/O-optimized Almacenamiento de Amazon DocumentDB: diseñado para clientes que priorizan la previsibilidad de los precios o que tienen I/O aplicaciones intensivas. La I/O-optimized configuración ofrece un rendimiento mejorado, un mayor rendimiento y una latencia reducida para los clientes con cargas de trabajo I/O intensivas. Si espera que sus I/O costos superen el 25%

de los costos totales de los clústeres de Amazon DocumentDB, esta opción ofrece una mejor relación precio-rendimiento. Con la configuración de I/O-optimized almacenamiento de Amazon DocumentDB, no se le cobrará en función de I/O las operaciones, lo que garantiza unos costes predecibles en cada ciclo de facturación. La configuración estabiliza los costos y, al mismo tiempo, mejora el rendimiento.

Puede cambiar los clústeres de bases de datos existentes una vez cada 30 días al almacenamiento de Amazon DocumentDB. I/O-optimized Puede volver al almacenamiento estándar de Amazon DocumentDB en cualquier momento. La próxima fecha para modificar la configuración de almacenamiento se I/O-optimized puede rastrear con el `describe-db-clusters` comando mediante la página de configuración del Consola de administración de AWS clúster AWS CLI o a través de ella.

Puede crear un nuevo clúster de bases de datos que incluya la I/O-optimized configuración de Amazon DocumentDB o convertir los clústeres de bases de datos existentes con unos pocos clics Consola de administración de AWS, un solo cambio de parámetro en AWS Command Line Interface (AWS CLI) o mediante los AWS SDK. No es necesario interrumpir ni reiniciar las instancias durante o después de modificar la configuración de almacenamiento.

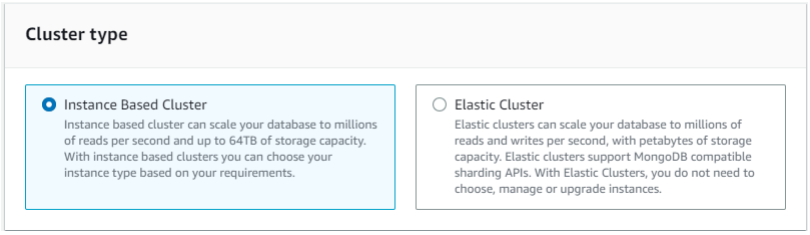


Creación de un clúster I/O-optimized

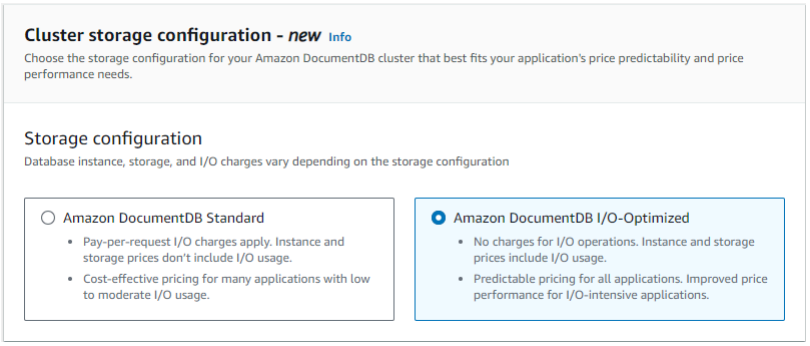
Using the Consola de administración de AWS

Para crear o modificar un I/O-optimized clúster mediante Consola de administración de AWS:

1. En la consola de administración de Amazon DocumentDB, en Clústeres, elija Crear o seleccione el clúster, elija Acciones y, a continuación, elija Modificar.
2. Si va a crear un clúster nuevo, asegúrese de elegir Clúster basado en instancias en la sección Tipo de clúster (esta es la opción predeterminada).



3. En la sección Configuración, en Configuración de almacenamiento en clústeres, elija Amazon DocumentDB. I/O-Optimized



4. Complete la creación o modificación del clúster y elija Crear clúster o Modificar clúster.

   Para ver el proceso completo de creación de un clúster, consulte Crear un clúster y una instancia principal mediante el Consola de administración de AWS.

   Para ver el proceso completo de modificación de un clúster, consulte Modificación de un clúster de Amazon DocumentDB.

Using the AWS CLI

Para crear un I/O-optimized clúster mediante: AWS CLI

En los siguientes ejemplos, sustituya cada uno *user input placeholder* por su propia información.

Para 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
```

Para 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
```

Análisis de costos para determinar la configuración del almacenamiento

Con Amazon DocumentDB, tiene la flexibilidad de elegir la configuración de almacenamiento para cada clúster de bases de datos que tenga. Para poder asignar correctamente los clústeres entre estándar y I/O-optimized, puede realizar un seguimiento de los costos de Amazon DocumentDB agrupados en clústeres. Para ello, puede agregar etiquetas a los clústeres existentes, habilitar el etiquetado de asignación de costos en el panel de Administración de facturación y costos de AWS y analizar los costos de un clúster determinado en el AWS Cost Explorer Service. Para obtener información sobre el análisis de costos, consulte nuestro blog Using cost allocation tags.

Determinación del estado de un clúster

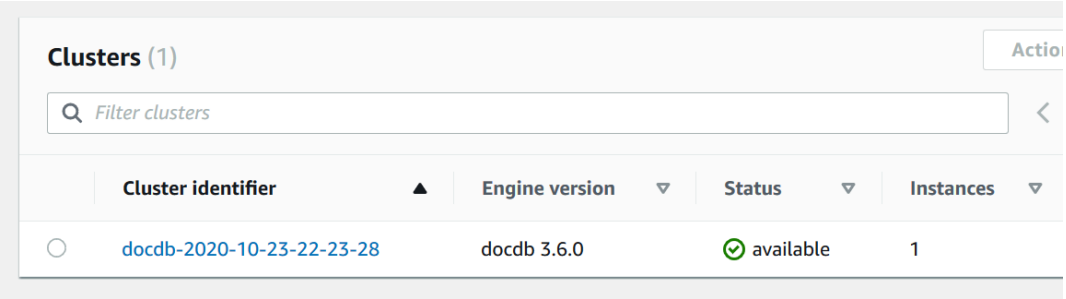
Puede determinar el estado de un clúster mediante el comando o. Consola de administración de AWS AWS CLI

Using the Consola de administración de AWS

Utilice el procedimiento siguiente para ver el estado de su clúster de Amazon DocumentDB mediante el Consola de administración de AWS

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb

2. En el panel de navegación, seleccione Clusters (Clústeres).
3. En la columna Cluster identifier (Identificador de clúster), busque el nombre del clúster que le interese. A continuación, para encontrar el estado del clúster, consulte en esa fila la columna Status (Estado), tal y como se muestra a continuación.



Using the AWS CLI

Utilice la operación `describe-db-clusters` para ver el estado del clúster de Amazon DocumentDB mediante la AWS CLI.

El siguiente código busca el estado del clúster `sample-cluster`.

Para Linux, macOS o Unix:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster  \
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Para Windows:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster  ^
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
[
    [
        "sample-cluster",
        "available"
    ]
]
```

Ciclo de vida del clúster de Amazon DocumentDB

El ciclo de vida de un clúster de Amazon DocumentDB incluye crear, describir, modificar y eliminar el clúster. En esta sección se proporciona información sobre cómo realizar estos procesos.

Temas

- Creación de un clúster de Amazon DocumentDB
- Descripción de los clústeres de Amazon DocumentDB
- Modificación de un clúster de Amazon DocumentDB
- Determinación de las tareas de mantenimiento pendientes
- Actualización de un parche a la versión del motor de un clúster
- Actualizar el sistema operativo de un clúster
- Detención e inicio de un clúster de bases de datos de Amazon DocumentDB
- Eliminación de un clúster de Amazon DocumentDB

Creación de un clúster de Amazon DocumentDB

Un clúster de Amazon DocumentDB consta de instancias y de un volumen de clúster que representa los datos del clúster. El volumen de clúster se replica de seis formas en tres zonas de disponibilidad como un único volumen virtual. El clúster contiene una instancia principal y, opcionalmente, hasta 15 instancias de réplica.

En las siguientes secciones se muestra cómo crear un clúster de Amazon DocumentDB utilizando el Consola de administración de AWS o el. AWS CLI A continuación, puede añadir instancias de réplica al clúster. Si utiliza la consola para crear el clúster de Amazon DocumentDB, se crea automáticamente una instancia principal al mismo tiempo. Si utiliza el AWS CLI para crear su clúster de Amazon DocumentDB, una vez que el estado del clúster esté disponible, debe crear la instancia principal para ese clúster.

Requisitos previos

A continuación, se describen los requisitos previos para crear un clúster de Amazon DocumentDB.

Si no tiene uno Cuenta de AWS, complete los siguientes pasos para crear uno.

Para inscribirse en un Cuenta de AWS

1. https://signin.aws.amazon.com/signup¿Abierto? request_type=registrarse.

2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica o mensaje de texto e indicar un código de verificación en el teclado del teléfono.

   Cuando te registras en un, se crea un Cuenta de AWS. Usuario raíz de la cuenta de AWS El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar Tareas que requieren acceso de usuario raíz.

Requisitos previos de VPC

Solo puede crear un clúster de Amazon DocumentDB en una instancia de Amazon Virtual Private Cloud (Amazon VPC). Amazon VPC debe tener como mínimo una subred en al menos dos de las zonas de disponibilidad para que la use con un clúster de base de datos de Amazon DocumentDB. Mediante la distribución de las instancias del clúster en las zonas de disponibilidad, se garantiza que habrá instancias disponibles en el clúster en el caso improbable de que se produzca un error en una zona.

Requisitos previos de la subred

Al crear un clúster de Amazon DocumentDB, debe elegir una VPC y el grupo de subredes correspondiente dentro de la VPC para lanzar el clúster. Las subredes determinan la zona de disponibilidad y el rango de direcciones IP de esa zona de disponibilidad que se quieran utilizar para lanzar una instancia. Para los fines de esta exposición, usaremos los términos subred y zona de disponibilidad indistintamente. Un grupo de subredes es un conjunto de subredes (o zonas de disponibilidad). Un grupo de subredes le permite especificar las zonas de disponibilidad que desea utilizar para lanzar instancias de Amazon DocumentDB. Por ejemplo, para ofrecer un alto nivel de disponibilidad en un clúster con tres instancias, se recomienda que cada una de dichas instancias esté aprovisionada en una zona de disponibilidad distinta. De esa forma, si una de las zonas de disponibilidad deja de funcionar, solo se verá afectada una sola instancia.

Las instancias de Amazon DocumentDB actualmente pueden aprovisionarse en hasta tres zonas de disponibilidad. Aunque un grupo de subredes tenga más de tres subredes, solo podrá utilizar tres de esas subredes para crear un clúster de Amazon DocumentDB. Por ello, es aconsejable que al crear un grupo de subredes solo elija las tres subredes en las que desee implementar las instancias. En el Este de EE. UU. (Norte de Virginia), el grupo de subredes puede tener seis subredes (o zonas de disponibilidad). Sin embargo, cuando se aprovisione un clúster de Amazon DocumentDB, Amazon DocumentDB elige tres de estas zonas de disponibilidad que utiliza para aprovisionar instancias.

Por ejemplo, supongamos que al crear un clúster, Amazon DocumentDB elige las zonas de disponibilidad {1A, 1B y 1C}. Si intenta crear una instancia en la zona de disponibilidad {1D}, la llamada a la API no funcionará correctamente. Sin embargo, si decide crear una instancia sin especificar la zona de disponibilidad concreta, Amazon DocumentDB elegirá una zona de disponibilidad por usted. Amazon DocumentDB utiliza un algoritmo para equilibrar la carga de las instancias en las zonas de disponibilidad para ayudarle a obtener una alta disponibilidad. Por ejemplo, si se aprovisionan tres instancias, de forma predeterminada se repartirán entre tres zonas de disponibilidad en lugar de aprovisionar todas ellas en una única zona de disponibilidad.

Recomendaciones:

- A menos que tenga un motivo específico, cree siempre un grupo de subredes con tres subredes. De esta forma, se asegurará de que los clústeres con tres o más instancias puedan lograr una mayor disponibilidad, debido a que las instancias se aprovisionarán en tres zonas de disponibilidad distintas.
- Reparta siempre las instancias en varias zonas de disponibilidad para lograr una alta disponibilidad. Nunca ponga todas las instancias de un clúster en una única zona de disponibilidad.
- Debido a que pueden producirse eventos de conmutación por error en cualquier momento, no debe dar por hecho que una instancia principal o las instancias de réplica siempre están en una zona de disponibilidad determinada.

Requisitos previos adicionales

A continuación se indican algunos requisitos adicionales para crear un clúster de Amazon DocumentDB:

- Si se conecta AWS utilizando credenciales AWS Identity and Access Management (IAM), su cuenta de IAM debe tener políticas de IAM que otorguen los permisos necesarios para realizar las operaciones de Amazon DocumentDB.

  Si utiliza una cuenta de IAM para acceder a la consola de Amazon DocumentDB, primero debe iniciar sesión en ella con su cuenta de IAM. Consola de administración de AWS A continuación, vaya a la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
- Si desea adaptar los parámetros de configuración a un clúster, debe especificar un grupo de parámetros de clúster y un grupo de parámetros con la configuración de parámetros necesaria. Para obtener información acerca de cómo crear o modificar un grupo de parámetros de clúster o

un grupo de parámetros de base de datos, consulte Administración de los grupos de parámetros de clúster de Amazon DocumentDB.

- Debe determinar el número de TCP/IP puerto que desea especificar para su clúster. Los firewalls de algunas compañías bloquean las conexiones a estos puertos predeterminados para Amazon DocumentDB. Si el firewall de su compañía bloquea el puerto predeterminado, elija otro puerto para el clúster. Todas las instancias de un clúster usan el mismo puerto.

Crear un clúster y una instancia principal mediante el Consola de administración de AWS

Los siguientes procedimientos describen cómo se usa la consola para lanzar un clúster de Amazon DocumentDB con una o varias instancias.

Creación de un clúster: uso de la configuración predeterminada

Para crear un clúster con instancias que utilicen la configuración predeterminada, utilice el Consola de administración de AWS

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. Si desea crear su clúster en una región que no Región de AWS sea la de EE. UU. Este (Virginia del Norte), elija la región en la lista de la sección superior derecha de la consola.
3. En el panel de navegación, elija Clusters (Clústeres) y, a continuación, elija Create (Crear).

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

4. En la página Crear clúster de Amazon DocumentDB, complete el panel de Configuración.

   a. Identificador de clúster: acepte el nombre que ha proporcionado Amazon DocumentDB o introduzca un nombre para el clúster; por ejemplo, **`sample-cluster`**.

      Restricciones en cuanto a la nomenclatura de los clústeres:

      - Debe tener [1-63] letras, números o guiones.
      - El primer carácter debe ser una letra.

- No puede terminar con un guion ni contener dos guiones consecutivos.
- Debe ser único para todos los clústeres de Amazon RDS, Neptune y Amazon DocumentDB por región. Cuenta de AWS

b. Versión del motor: acepte la versión predeterminada del motor 5.0.0 o, si lo desea, elija 8.0.0, 4.0.0 o 3.6.0.

c. Clase de instancia: puede aceptar la `db.r5.large` predeterminada o elegir una en la lista desplegable.

d. Número de instancias: en la lista, elija el número de instancias que desee crear cuando se restaure el clúster. La primera es la instancia principal, todas las demás instancias son instancias de réplica de solo lectura. Puede añadir y eliminar instancias más adelante si las necesita. De forma predeterminada, un clúster de Amazon DocumentDB se lanzará con tres instancias (una principal y dos réplicas).

5. Complete la sección Configuración del almacenamiento en clúster.

   Elija Amazon DocumentDB Standard (predeterminado) o Amazon DocumentDB. I/O-Optimized Para obtener más información, consulte Configuraciones de almacenamiento en clústeres de Amazon DocumentDB.

6. Rellene el panel Authentication (Autenticación).

   a. Nombre de usuario: ingrese un nombre para el usuario principal. Para iniciar sesión en el clúster, debe utilizar el nombre de usuario principal.

      Restricciones de denominación de usuario principal:

      - La longitud es de [1-63] caracteres alfanuméricos.
      - El primer carácter debe ser una letra.
      - No puede ser una palabra reservada por el motor de base de datos.

   b. Seleccione una de las siguientes opciones de contraseña:

      - Administrado en AWS Secrets Manager: elija esta opción si desea AWS Secrets Manager administrar automáticamente su contraseña de usuario principal.

        Si elige esta opción, configure la clave de KMS. Para ello, cree la suya propia o utilice una clave creada por Secrets Manager.
      - Autoadministrado: elija esta opción si desea autoadministrar la contraseña de su usuario principal. Si elige esta opción, introduzca una contraseña para el usuario principal y

confírmela. Para iniciar sesión en el clúster, debe utilizar la contraseña del usuario principal.

Restricciones para la contraseña:

- La longitud es de [8-100] caracteres ASCII imprimibles.
- Se puede utilizar cualquier carácter ASCII imprimible, excepto los siguientes:
  - **/** (barra inclinada)
  - **"** (comillas dobles)
  - **@** (símbolo de arroba)

7. En la parte inferior de la pantalla, elija una de las opciones siguientes:
   - Para crear el clúster ahora, elija Create cluster (Crear clúster).
   - Si no desea crear el clúster, elija Cancel (Cancelar).
   - Para configurar el clúster antes de crearlo, elija Show additional configurations (Mostrar configuraciones adicionales) y, a continuación, continúe en Creación de un clúster: configuraciones adicionales.

     Las configuraciones que aparecen en Additional Configurations (Configuraciones adicionales) son las siguientes:

     - Configuración de red: el valor predeterminado es utilizar el grupo de seguridad de VPC `default`.
     - Opciones de clúster: de forma predeterminada, se utilizan el puerto 27017 y el grupo de parámetros predeterminado.
     - Cifrado: el valor predeterminado es habilitar el cifrado con la clave `(default) aws/rds`.

       > **Important**
       >
       > Después de que se cifra un clúster, no es posible descifrarlo.

     - Copia de seguridad: el valor predeterminado es conservar las copias de seguridad durante 1 día y dejar que Amazon DocumentDB seleccione el periodo de copia de seguridad.
     - Exportaciones de registros: de forma predeterminada, no se exportan los registros de auditoría a CloudWatch Logs.

- Mantenimiento: de forma predeterminada, Amazon DocumentDB elige el período de mantenimiento.
- Protección contra la eliminación: proteja sus clústeres contra la eliminación accidental. El valor predeterminado para el clúster creado con la consola es enabled (habilitado).

Si acepta los valores predeterminados ahora, podrá cambiar la mayoría de ellos más adelante modificando el clúster.

8. Habilite la conexión entrante del grupo de seguridad de su clúster.

   Si no cambiaste la configuración predeterminada de tu clúster, creaste un clúster con el grupo de seguridad predeterminado para la VPC predeterminada de la región en cuestión. Para conectarse a Amazon DocumentDB, tiene que habilitar las conexiones entrantes en el puerto 27017 (o el puerto que prefiera) para el grupo de seguridad de su clúster.

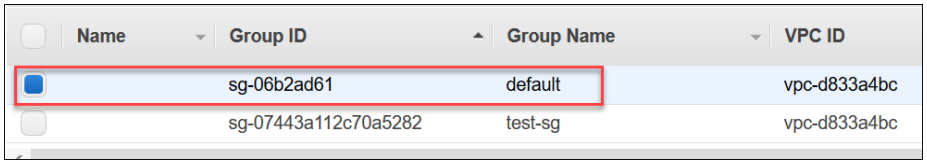
   Cómo añadir una conexión entrante al grupo de seguridad de su clúster

   a. Inicie sesión en Consola de administración de AWS y abra la consola de Amazon EC2 en. https://console.aws.amazon.com/ec2/

   b. En la sección Resources (Recursos) de la ventana principal, seleccione Security groups (Grupos de seguridad).



   c. En la lista de grupos de seguridad, localice el grupo de seguridad que utilizó para crear el clúster (lo más probable es que fuera el grupo de seguridad predeterminado) y seleccione la casilla situada a la izquierda del nombre del grupo de seguridad.



   d. En el menú Actions (Acciones), elija Edit inbound rules (Editar reglas entrantes) y, a continuación, seleccione o especifica las restricciones de las reglas.

i. Tipo: en la lista, elija el protocolo para abrir al tráfico de la red.

ii. Protocolo: en la lista, elija el tipo de protocolo.

iii. Intervalo de puertos: para una regla personalizada, introduzca un número de puerto o un intervalo de puertos. Asegúrese de que el número de puerto o el rango de puertos incluye el puerto que especificó cuando creó el clúster (predeterminado: 27017).

iv. Origen: especifica el tráfico que puede llegar a su instancia. En la lista, elija el origen del tráfico. Si elige Personalizado, especifique una dirección IP única o un intervalo de direcciones IP en notación CIDR (por ejemplo, 203.0.113). 5/32).

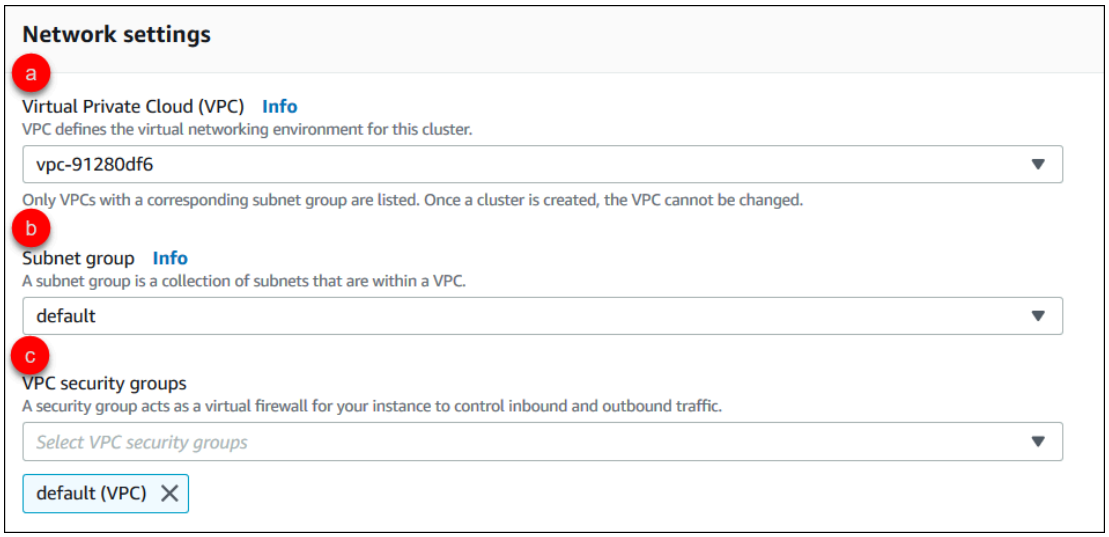
v. Descripción: escriba una descripción para la regla.

vi. Cuando finalice la creación de la regla, seleccione Save (Guardar).

Creación de un clúster: configuraciones adicionales

Si desea aceptar los valores predeterminados para el clúster, puede omitir los pasos que se describen a continuación y elegir Create cluster (Crear clúster).

1. Rellene el panel Network settings (Configuración de red).

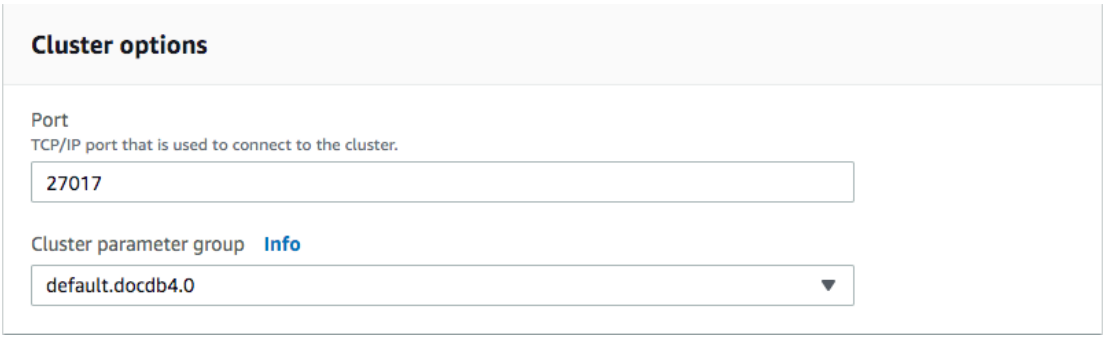


a. Nube privada virtual (VPC): en la lista, elija la Amazon VPC en la que desea lanzar este clúster.

b. Grupo de subredes: en la lista, elija el grupo de subredes que desea utilizar para este clúster.

c. Grupos de seguridad de VPC: en la lista, elija el grupo de seguridad para el clúster.

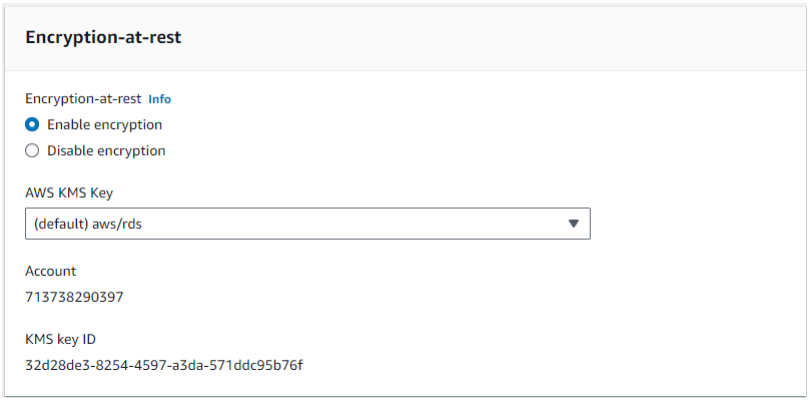
2. Rellene el panel Cluster options (Opciones de clúster).



a. Puerto de base de datos: utilice las flechas hacia arriba y hacia abajo para configurar el TCP/IP puerto que usarán las aplicaciones para conectarse a la instancia.

b. Grupo de parámetros de clúster: en la lista de grupos de parámetros, elija el grupo de parámetros de clúster para este clúster.

3. Rellene el panel Encryption (Cifrado).



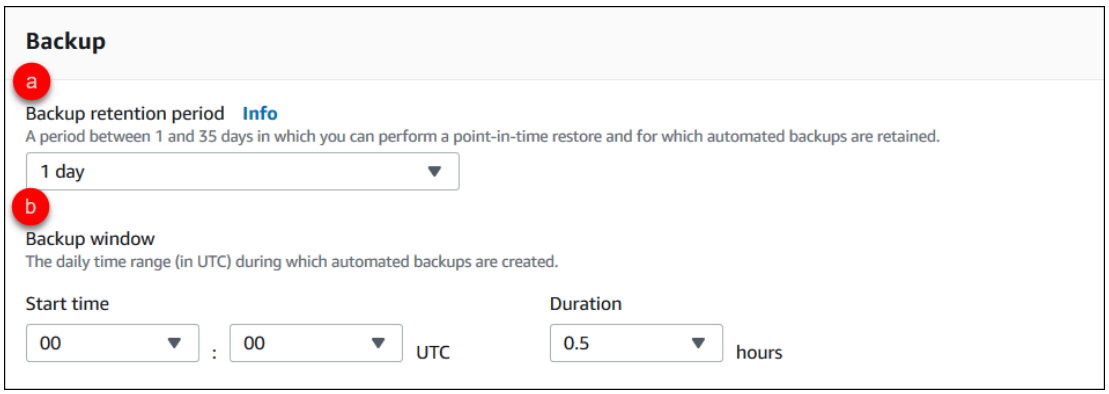
a. Encryption-at-rest—Elige una de las siguientes opciones:

   - Habilitar el cifrado: de manera predeterminada. Se cifran todos los datos en reposo. Si decide cifrar los datos, no puede deshacer esta acción.
   - Deshabilitar cifrado: los datos no se cifran.

b. AWS Clave KMS: solo está disponible si está cifrando sus datos. En la lista, elija la clave que desee utilizar para cifrar los datos de este clúster. El valor predeterminado es `(default) aws/rds`.

   Si ha elegido, Enter a key ARN (Escribir un ARN de clave), tendrá que introducir un nombre de recurso de Amazon (ARN) para la clave.

4. Rellene el panel Backup (Copia de seguridad).



a. Periodo de retención de copias de seguridad: en la lista, elija el número de días que se deben conservar las copias de seguridad automáticas de este clúster antes de eliminarlas.

b. Periodo de copia de seguridad: especifique la hora del día en que Amazon DocumentDB debe hacer las copias de seguridad de este clúster y la duración de las mismas.

   i. Hora de inicio: en la primera lista, elija la hora de inicio (UTC) para las copias de seguridad automáticas. En la segunda lista, elija el minuto de la hora en que desea que comiencen las copias de seguridad automáticas.

   ii. Duración: en la lista, elija el número de horas que se deben asignar para crear las copias de seguridad automáticas.

5. Complete el panel de exportación de registros seleccionando los tipos de registros que desea exportar a CloudWatch Logs.

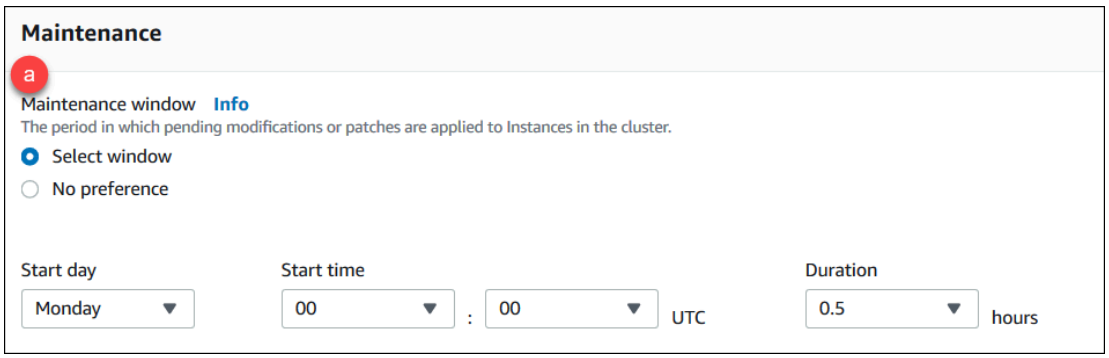


- Registros de auditoría: seleccione esta opción para permitir la exportación de registros de auditoría a Amazon CloudWatch Logs. Si selecciona Audit logs (Registros de auditoría), debe habilitar `audit_logs` en el grupo de parámetros personalizado del clúster. Para obtener más información, consulte Auditoría de eventos de Amazon DocumentDB.
- Registros del generador de perfiles: seleccione esta opción para permitir la exportación de los registros del generador de perfiles de operaciones a Amazon Logs. CloudWatch Si selecciona Profiler logs (Registros del generador de perfiles), también debe modificar los siguientes parámetros en el grupo de parámetros personalizado del clúster:
  - `profiler` establezca en `enabled`.

- `profiler_threshold_ms` establezca en un valor `[0-INT_MAX]` para definir el umbral para las operaciones de creación de perfiles.
- `profiler_sampling_rate` establezca en un valor `[0.0-1.0]` para establecer el porcentaje de operaciones lentas en el perfil.

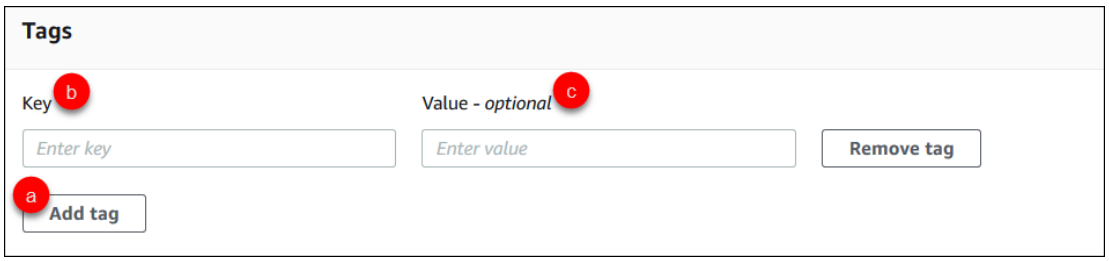
Para obtener más información, consulte Creación de perfiles de operaciones en Amazon DocumentDB.

6. Rellene el panel Maintenance (Mantenimiento).



- Elija una de las opciones siguientes.
  - Seleccionar periodo: permite especificar el día de la semana y la hora de inicio UTC en que Amazon DocumentDB debe realizar el mantenimiento del clúster y la duración de esta operación.
    a. Día de inicio: en la lista, elija el día de la semana en que debe iniciarse el mantenimiento del clúster.
    b. Hora de inicio: en las listas, elija la hora UTC y el minuto UTC en que debe iniciarse el mantenimiento.
    c. Duración: en la lista, elija el tiempo que se debe asignar para mantenimiento del clúster. Si el mantenimiento no puede finalizar en el tiempo especificado, el proceso de mantenimiento continuará pasado ese tiempo hasta que finalice.
  - Sin preferencia: Amazon DocumentDB selecciona el día de la semana, la hora de inicio y la duración para realizar el mantenimiento.

7. Si desea añadir una o más etiquetas a este clúster, complete el panel Tags (Etiquetas).



Para cada etiqueta que desee añadir al clúster, repita los pasos siguientes. Puede disponer de hasta 10 en un clúster.

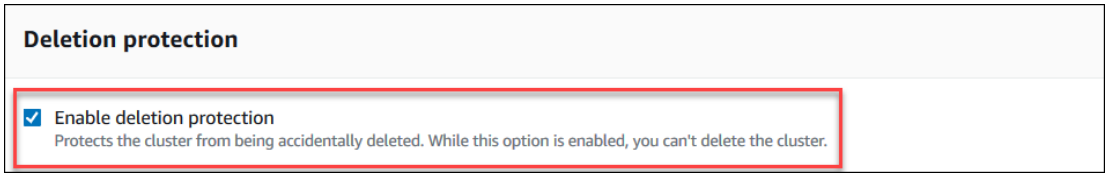
a. Elija Add tags (Añadir etiquetas).

b. Escriba la clave de la etiqueta.

c. Si lo desea, escriba el valor de la etiqueta.

Para quitar una etiqueta, elija Remove tag (Eliminar etiqueta).

8. La protección contra eliminación se habilita de forma predeterminada cuando crea un clúster mediante la consola. Para deshabilitar la protección contra eliminación, desactive Habilitar la protección contra eliminación. Cuando esté habilitada, la protección contra eliminación evitará que se elimine un clúster. Para quitar un clúster protegido contra eliminación, en primer lugar debe modificar el clúster para deshabilitar la protección contra eliminación.



Para obtener más información sobre la protección contra eliminación, consulte Eliminación de un clúster de Amazon DocumentDB.

9. Para crear el clúster, elija Create cluster (Crear clúster). De lo contrario, seleccione Cancelar.

Crear un clúster con el AWS CLI

Los procedimientos siguientes describen cómo utilizarla AWS CLI para lanzar un clúster de Amazon DocumentDB y crear una réplica de Amazon DocumentDB.

Parameters

- **`--db-cluster-identifier`**: obligatorio. Una cadena en minúsculas que identifica este clúster.

Restricciones en cuanto a la nomenclatura de los clústeres:

- Debe tener [1-63] letras, números o guiones.
- El primer carácter debe ser una letra.
- No puede terminar con un guion ni contener dos guiones consecutivos.
- Debe ser único para todos los clústeres (en Amazon RDS, Amazon Neptune y Amazon DocumentDB) por AWS cuenta y región.

- **`--engine`**: obligatorio. Debe ser **`docdb`**.
- **`--deletion-protection | --no-deletion-protection`**: opcional. Cuando la protección contra eliminación esté habilitada, evitará que se elimine un clúster. Cuando usa el AWS CLI, la configuración predeterminada es tener deshabilitada la protección contra la eliminación.

  Para obtener más información sobre la protección contra eliminación, consulte Eliminación de un clúster de Amazon DocumentDB.
- **`--storage-type standard | iopt1`**: opcional. Predeterminado: **`standard`**. Configuración de almacenamiento en clústeres. Los valores válidos son `standard` (Estándar) o `iopt1` (I/O-optimized).
- **`--master-username`**: obligatorio. El nombre de usuario que se utiliza para autenticar al usuario.

  Restricciones en cuanto a la nomenclatura de los usuarios maestros:

  - La longitud es de [1-63] caracteres alfanuméricos.
  - El primer carácter debe ser una letra.
  - No puede ser una palabra reservada por el motor de base de datos.
- **`--master-user-password`**: opcional. La contraseña de usuario que se utiliza para autenticar al usuario.

  Restricciones para la contraseña maestra:

  - La longitud es de [8-100] caracteres ASCII imprimibles.
  - Se puede utilizar cualquier carácter ASCII imprimible, excepto los siguientes:
    - **`/`** (barra inclinada)

  - **"** (comillas dobles)
  - **@** (símbolo de arroba)

- **`--manage-master-user-password`**: opcional. Amazon DocumentDB genera la contraseña del usuario maestro y la administra durante todo su ciclo de vida en Secrets Manager.

Para ver parámetros adicionales, consulte https://docs.aws.amazon.com/documentdb/latest/APIReference/API_CreateDBCluster.html CreateDBCluster.

Para lanzar un clúster de Amazon DocumentDB mediante AWS CLI

Para crear un clúster de Amazon DocumentDB, llame al. `create-db-cluster` AWS CLI El siguiente AWS CLI comando crea un clúster de Amazon DocumentDB denominado `sample-cluster` con la protección contra borrados habilitada. Para obtener más información sobre la protección contra eliminación, consulte Eliminación de un clúster de Amazon DocumentDB.

Además, `--engine-version` es un parámetro opcional que utiliza de forma predeterminada la última versión principal del motor. La versión predeterminada actual del motor es la 5.0.0 (nota: Amazon DocumentDB 8.0 está disponible, pero debe especificarse de forma explícita). Cuando se publiquen nuevas versiones principales del motor, la versión del motor predeterminada para `--engine-version` se actualizará para reflejar la última versión principal del motor. Como resultado, para las cargas de trabajo de producción, y especialmente las que dependen de la creación de scripts, la automatización o CloudFormation las plantillas, especifique explícitamente la versión principal `--engine-version` deseada.

> **Note**
>
> Si no se especifica un `db-subnet-group-name` o un `vpc-security-group-id`, Amazon DocumentDB utilizará el grupo de subredes y el grupo de seguridad de Amazon VPC predeterminados para la región determinada.

Para 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
```

Para 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
```

La salida de esta operación será similar a lo que se indica a continuación (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 creación del clúster puede tardar varios minutos. Puedes usar el Consola de administración de AWS o AWS CLI para supervisar el estado de tu clúster. Para obtener más información, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

> **Important**
>
> Cuando lo utiliza AWS CLI para crear un clúster de Amazon DocumentDB, no se crea ninguna instancia. Por lo tanto, tendrá que crear de forma explícita una instancia principal y las instancias de réplica que necesite. Puede usar la consola o AWS CLI crear las instancias. Para obtener más información, consulte Agregación de una instancia de Amazon DocumentDB a un clúster.

Para obtener más información, consulte CreateDBCluster en la referencia de la API de Amazon DocumentDB.

Descripción de los clústeres de Amazon DocumentDB

Puede usar la consola de administración de Amazon DocumentDB o la AWS CLI para ver detalles como los puntos de conexión, los grupos de seguridad, las VPC y los grupos de parámetros relacionados con sus clústeres de Amazon DocumentDB.

Para obtener más información, consulte los siguientes temas:

- Supervisión del estado de un clúster de Amazon DocumentDB
- Búsqueda de puntos de conexión de un clúster

Using the Consola de administración de AWS

Use el siguiente procedimiento para ver los detalles de un clúster de Amazon DocumentDB especificado mediante la consola.

1. Inicie sesión en y abra la consola Consola de administración de AWS de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Clusters (Clústeres).

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En la lista de clústeres, elija el nombre del clúster cuyos detalles desea ver. La información sobre el clúster está organizada en las siguientes agrupaciones:

   - Resumen: información general sobre el clúster, incluida la versión del motor, el estado del clúster, el mantenimiento pendiente y el estado de su grupo de parámetros.
   - Conectividad y seguridad: la sección Conectar muestra los puntos de conexión para conectarse a este clúster con el intérprete de comandos de mongo o con una aplicación. La sección Grupos de seguridad muestra los grupos de seguridad asociados a este clúster y su ID de VPC y descripciones.
   - Configuración la sección Detalles del clúster muestra detalles sobre el clúster, incluidos el nombre de recurso de Amazon (ARN), el punto de conexión y el grupo de parámetros de clúster. También muestra la información de copia de seguridad del clúster, los detalles de mantenimiento y la configuración de seguridad y red. La sección Instancias del clúster muestra las instancias que pertenecen a este clúster con el rol de cada instancia y el estado del grupo de parámetros del clúster.
   - Supervisión: las métricas de Amazon CloudWatch Logs para este clúster. Para obtener más información, consulte Supervisión de Amazon DocumentDB con CloudWatch.
   - Eventos y etiquetas: en la sección de eventos recientes se enumeran los eventos recientes de este clúster. Amazon DocumentDB mantiene un registro de los eventos relacionados con los clústeres, las instancias, las instantáneas, los grupos de seguridad y los grupos de parámetros de clúster. Esta información incluye la fecha, la hora y el mensaje asociados a cada evento. La sección Etiquetas muestra las etiquetas asociadas a este clúster.

Using the AWS CLI

Para ver los detalles de sus clústeres de Amazon DocumentDB mediante AWS CLI, utilice el `describe-db-clusters` comando tal y como se muestra en los siguientes ejemplos. Para obtener más información, consulte la descripción de DBClusters en la referencia de la API de administración de recursos de Amazon DocumentDB.

> **Note**
>
> Para ciertas características de administración, como la administración del ciclo de vida de clúster y de instancia, Amazon DocumentDB aprovecha la tecnología operativa que se comparte con Amazon RDS. El parámetro de filtro `filterName=engine,Values=docdb` devuelve solo clústeres de Amazon DocumentDB.

Example

Ejemplo 1: enumerar todos los clústeres de Amazon DocumentDB

El siguiente AWS CLI código muestra los detalles de todos los clústeres de Amazon DocumentDB de una región.

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "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

Ejemplo 2: mostrar todos los detalles de un clúster de Amazon DocumentDB especificado

El siguiente AWS CLI código muestra los detalles del clústersample-cluster.

Para Linux, macOS o Unix:

```
aws docdb describe-db-clusters \
   --filter Name=engine,Values=docdb \
   --db-cluster-identifier sample-cluster
```

Para Windows:

```
aws docdb describe-db-clusters ^
   --filter Name=engine,Values=docdb ^
   --db-cluster-identifier sample-cluster
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "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

Ejemplo 3: mostrar detalles específicos de un clúster de Amazon DocumentDB

Para enumerar un subconjunto de los detalles de los clústeres mediante AWS CLI, agregue un `--query` que especifique qué miembros del clúster debe incluir la `describe-db-clusters` operación. El parámetro de `--db-cluster-identifier` es el identificador del clúster concreto del que desea mostrar los detalles. Para obtener más información sobre consultas, consulte

Cómo filtrar la salida con la opción `--query` en la Guía de usuario de AWS Command Line Interface .

En el siguiente ejemplo de la se muestran las instancias de un clúster de Amazon DocumentDB.

Para Linux, macOS o Unix:

```
aws docdb describe-db-clusters \
    --filter Name=engine,Values=docdb \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterMembers]'
```

Para Windows:

```
aws docdb describe-db-clusters ^
    --filter Name=engine,Values=docdb ^
    --db-cluster-identifier sample-cluster ^
    --query 'DBClusters[*].[DBClusterMembers]'
```

La salida de esta operación será similar a lo que se indica a continuación.

```
[
    [
        [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            },
            {
                "DBInstanceIdentifier": "sample-instance-2",
                "IsClusterWriter": false,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            }
        ]
    ]
]
```

Modificación de un clúster de Amazon DocumentDB

Para modificar un clúster, el clúster debe tener el estado available (disponible). No puede modificar un clúster que está detenido. Si el clúster está detenido, inicie primero el clúster, espere a que el clúster esté disponible y, a continuación, realice las modificaciones deseadas. Para obtener más información, consulte Detención e inicio de un clúster de bases de datos de Amazon DocumentDB.

Using the Consola de administración de AWS

Utilice el siguiente procedimiento para modificar un clúster de Amazon DocumentDB específico mediante la consola.

Modificación de un clúster de Amazon DocumentDB

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Clusters (Clústeres).

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. Especifique el clúster que desee modificar seleccionando el botón situado a la izquierda del nombre del clúster.
4. Elija Actions (Acciones) y después Modify (Modificar).
5. En el panel Modify Cluster: <cluster-name> (Modificar clúster: <nombre-clúster>, realice los cambios que desee. Puede realizar cambios en las siguientes áreas:

   - Especificaciones del clúster: nombre del clúster, grupos de seguridad y administración de credenciales.
   - Configuración del almacenamiento en clúster: el modo de almacenamiento de datos del clúster. Elija entre estándar y I/O-Optimized configuración.
   - Opciones del clúster: el grupo de parámetros y el puerto del clúster.
   - Copia de seguridad: el período de copia de seguridad y de retención de la copia de seguridad del clúster.

- Exportación de registros: permite habilitar o deshabilitar los registros de auditoría o perfiles de exportación.
- Mantenimiento: permite establecer la ventana de mantenimiento del clúster.
- Protección de eliminación: permite habilitar o deshabilitar la protección de eliminación en el clúster. La protección contra eliminación está habilitada de forma predeterminada.

6. Cuando haya terminado, elija Continuar para ver un resumen de los cambios.
7. Si está satisfecho con los cambios, puede elegir Modificar clúster para modificar el clúster. También puede elegir Atrás o Cancelar para editar o cancelar los cambios, respectivamente.

Los cambios pueden tardar unos minutos en aplicarse. Solo se puede usar el clúster cuando su estado sea available (disponible). Puede monitorizar el estado del clúster mediante la consola o la AWS CLI. Para obtener más información, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

Using the AWS CLI

Utilice la operación `modify-db-cluster` para modificar el clúster especificado mediante la AWS CLI. Para obtener más información, consulte ModifyDBCluster en la referencia de la API de Amazon DocumentDB.

Parameters

- **`--db-cluster-identifier`**: obligatorio. El identificador del clúster de Amazon DocumentDB que va a modificar.
- **`--backup-retention-period`**: opcional. El número de días durante los que se retienen las copias de seguridad automatizadas. Los valores válidos están comprendidos entre 1 y 35.
- **`--storage-type`**: opcional. Configuración de almacenamiento en clústeres. Los valores válidos son `standard` (Estándar) o (). `iopt1` I/O-optimized
- **`--db-cluster-parameter-group-name`**: opcional. El nombre del grupo de parámetros de clúster que se va a usar para el clúster.
- **`--manage-master-user-password`**: opcional. Amazon DocumentDB genera la contraseña del usuario maestro y la administra durante todo su ciclo de vida en Secrets Manager.
- **`--rotate-master-user-password`**: opcional. Secrets Manager genera una nueva versión de secreto para el secreto existente. La nueva versión del secreto contiene la nueva contraseña de usuario principal. Amazon DocumentDB cambia la contraseña del usuario principal para el clúster para que coincida con la contraseña de la nueva versión del secreto.

Debe especificar la opción **`--apply-immediately`** cuando rota la contraseña principal.

- **`--master-user-password`**: opcional. La nueva contraseña del usuario de la base de datos principal.

  Restricciones para la contraseña:

  - La longitud es de 8 a 100 caracteres ASCII imprimibles.
  - Se puede utilizar cualquier carácter ASCII imprimible, excepto los siguientes:
    - **`/`** (barra inclinada)
    - **`"`** (comillas dobles)
    - **`@`** (símbolo de arroba)
- **`--new-db-cluster-identifier`**: opcional. El nuevo identificador del clúster cuando se cambia el nombre de un clúster. Este valor se almacena como una cadena en minúsculas.

  Restricciones en la nomenclatura:

  - Debe tener [1-63] letras, números o guiones.
  - El primer carácter debe ser una letra.
  - No puede terminar con un guion ni contener dos guiones consecutivos.
  - Debe ser único para todos los clústeres de Amazon RDS, Amazon Neptune y Amazon DocumentDB por región Cuenta de AWS.
- **`--preferred-backup-window`**: opcional. El intervalo de tiempo diario (en Tiempo Universal Coordinado [UTC]) durante el cual se crean copias de seguridad automatizadas.
  - Formato: `hh24:mm-hh24:mm`
- **`--preferred-maintenance-window`**: opcional. El intervalo de tiempo semanal (en UTC) durante el cual puede llevarse a cabo el mantenimiento del sistema.
  - Formato: `ddd:hh24:mm-ddd:hh24:mm`
  - Días válidos: `Sun`, `Mon`, `Tue`, `Wed`, `Thu`, `Fri` y `Sat`.
- **`--deletion-protection`** o **`--no-deletion-protection`**: opcional. Si la protección contra eliminación se debe habilitar en este clúster. La protección contra eliminación evita que el clúster se elimine accidentalmente hasta que se modifique para deshabilitar la protección contra eliminación. Para obtener más información, consulte Eliminación de un clúster de Amazon DocumentDB.

- **`--apply-immediately`** o **`--no-apply-immediately`**: utilice `--apply-immediately` para aplicar los cambios inmediatamente. Utilice `--no-apply-immediately` para aplicar el cambio durante el próximo periodo de mantenimiento de su clúster.

Example

El siguiente código cambia el periodo de retención de copia de seguridad del clúster `sample-cluster`.

Para Linux, macOS o Unix:

```
aws docdb modify-db-cluster \
       --db-cluster-identifier sample-cluster \
       --apply-immediately \
       --backup-retention-period 7
```

Para Windows:

```
aws docdb modify-db-cluster ^
       --db-cluster-identifier sample-cluster ^
       --apply-immediately ^
       --backup-retention-period 7
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "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
    }
}
```

Los cambios pueden tardar unos minutos en aplicarse. Solo se puede usar el clúster cuando su estado sea available (disponible). Puede monitorizar el estado del clúster mediante la consola o la AWS CLI. Para obtener más información, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

Determinación de las tareas de mantenimiento pendientes

En la mayoría de los casos, el mantenimiento implica la actualización de los siguientes recursos del clúster:

- Sistema operativo (SO) de las instancias del clúster
- Versión del motor del clúster

Para obtener más información sobre el mantenimiento de Amazon DocumentDB, consulteMantenimiento de Amazon DocumentDB.

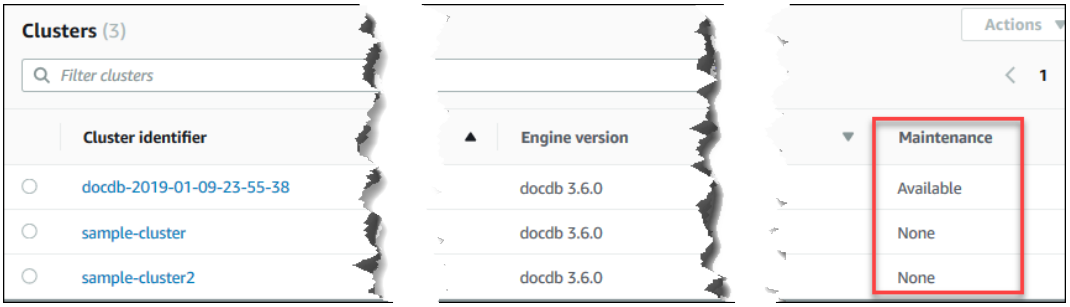
Using the Consola de administración de AWS

Puede utilizarla Consola de administración de AWS para determinar si un clúster tiene un mantenimiento pendiente.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Clusters (Clústeres).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

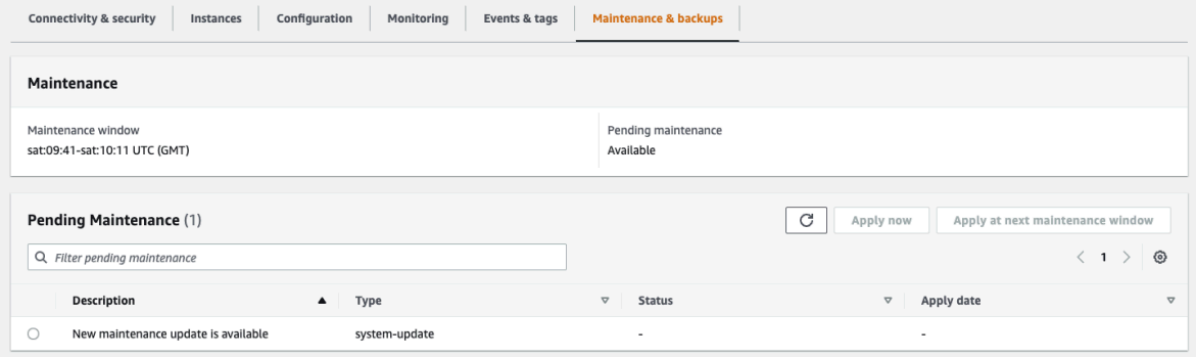
3. Localice la columna Maintenance (Mantenimiento) para determinar si un clúster tiene tareas de mantenimiento pendientes.



Ninguno indica que el clúster esté ejecutando la última versión del parche del motor y que sus instancias tengan las últimas actualizaciones del sistema operativo. La opción

Disponible indica que el clúster tiene un mantenimiento pendiente, lo que puede significar que es necesaria una actualización del motor o del sistema operativo.

4. Si su clúster tiene un mantenimiento pendiente, puede ver las acciones de mantenimiento pendientes seleccionando el nombre del clúster y seleccionando la pestaña Mantenimiento y copias de seguridad. La sección Mantenimiento pendiente muestra las acciones de mantenimiento pendientes.



Using the AWS CLI

Puede utilizarla AWS CLI para determinar si un clúster tiene la última versión del motor y las actualizaciones del sistema operativo más recientes mediante la `describe-pending-maintenance-actions` operación con los siguientes parámetros.

Parameters

- **`--resource-identifier`**: opcional. El ARN del recurso (clúster). Si este parámetro se omite, se enumeran las operaciones de mantenimiento pendientes de todos los clústeres.
- **`--region`**: opcional. La región de AWS en la que desea ejecutar esta operación, por ejemplo, `us-east-1`.

Example

Para 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
```

Para Windows:

```
aws docdb describe-pending-maintenance-actions ^
   --resource-identifier arn:aws:rds:aa-example-1:111122223333:cluster:sample-
cluster ^
   --region aa-example-1
```

La salida de esta operación será similar a lo que se indica a continuación.

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

Los siguientes tipos de acciones de mantenimiento se aplican a los clústeres de Amazon DocumentDB:

- `system-update`— Actualice el parche del motor para el clúster de Amazon DocumentDB.

  Si su clúster tiene disponible una actualización pendiente de un parche de motor, continúe con los pasos que se indican enActualización de un parche a la versión del motor de un clúster.
- `os-upgrade`— Actualice los sistemas operativos de todas las instancias de base de datos del clúster de Amazon DocumentDB mediante actualizaciones sucesivas.

Para obtener más información, consulte Actualizaciones del sistema operativo de Amazon DocumentDB.

Actualización de un parche a la versión del motor de un clúster

En esta sección, explicaremos cómo implementar una actualización de parches mediante el Consola de administración de AWS o el AWS CLI. Una actualización de parche es una actualización dentro de la misma versión del motor (por ejemplo, la actualización de una versión del motor 3.6 a una versión del motor 3.6 más reciente). Puede actualizarla inmediatamente o durante el próximo periodo de mantenimiento del clúster. Para determinar si el motor necesita una actualización, consulte Determinación de las tareas de mantenimiento pendientes. Tenga en cuenta que, al aplicar la actualización, el clúster sufrirá algún tiempo de inactividad.

> **Note**
>
> Si estás intentando actualizar una versión principal del motor a otra, como la 3.6 a la 5.0 o la 5.0 a la 8.0, consulta Actualización local de la versión principal Amazon DocumentDB oActualización del clúster de Amazon DocumentDB mediante AWS Database Migration Service.

Existen dos requisitos de configuración para obtener las actualizaciones de parches más recientes para la versión del motor de un clúster:

- El estado del clúster debe ser disponible.
- El clúster debe estar ejecutando una versión del motor más antigua.

Using the Consola de administración de AWS

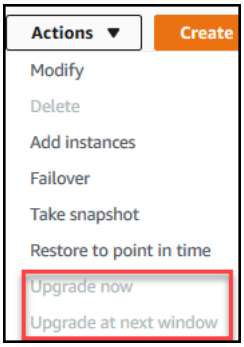
El siguiente procedimiento actualiza los parches de la versión del motor de su clúster a la versión más reciente con la consola. Puede actualizarla inmediatamente o durante el próximo periodo de mantenimiento del clúster.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.

2. En el panel de navegación, seleccione Clusters (Clústeres). En la lista de clústeres, elija el botón situado a la izquierda del clúster que desee actualizar. El estado del clúster debe ser disponible.

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En el menú Actions (Acciones), elija una de las opciones siguientes. Estas opciones de menú solo se pueden seleccionar si en el clúster elegido no se está ejecutando la última versión del motor.



- Actualizar ahora: inicia inmediatamente el proceso de actualización. El clúster se quedará sin conexión durante un tiempo mientras se actualiza a la última versión del motor.
- Actualizar en el siguiente periodo: inicia el proceso de actualización durante el próximo periodo de mantenimiento del clúster. El clúster se quedará sin conexión durante un tiempo mientras se actualiza a la última versión del motor.

4. Cuando se abra la ventana de confirmación, seleccione una de las siguientes opciones:
   - Actualizar: para actualizar el clúster a la última versión del motor de acuerdo con el plan elegido en el paso anterior.
   - Cancelar: para cancelar la actualización del motor del clúster y continuar con la versión actual del motor del clúster.

Using the AWS CLI

Puede aplicar actualizaciones de parches a su clúster mediante la `apply-pending-maintenance-action` operación AWS CLI and con los siguientes parámetros.

Parameters

- **`--resource-identifier`**: obligatorio. El ARN del clúster de Amazon DocumentDB que va a actualizar.
- **`--apply-action`**: obligatorio. `system-update`Utilícela para aplicar una actualización de parches del motor al clúster.
- **`--opt-in-type`**: obligatorio. Se permiten los siguientes valores.
  - `immediate`: aplicar inmediatamente la acción de mantenimiento.
  - `next-maintenance`: aplicar la operación de mantenimiento durante el siguiente período de mantenimiento.
  - `undo-opt-in`: cancelar todas las solicitudes de alta `next-maintenance` existentes.

Example

En el siguiente ejemplo, se aplica la actualización del parche del motor a`sample-cluster`.

Para 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
```

Para 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
```

El resultado de esta operación será similar a lo que se indica a continuación:

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

Actualizar el sistema operativo de un clúster

Puedes aplicar actualizaciones del sistema operativo (SO) en todas las instancias de tu clúster mediante la acción de mantenimiento a nivel de clústeros`os-upgrade`. Amazon DocumentDB actualiza las instancias de forma continua, unas cuantas a la vez, y actualiza la instancia principal en último lugar para minimizar las conmutaciones por error. La actualización se ejecuta durante el período de mantenimiento del clúster. Cuando una instancia recibe una actualización del sistema operativo, su caché de búfer comienza a vaciarse y las consultas de esa instancia pueden experimentar una latencia más alta hasta que el conjunto de trabajo se rellene desde el volumen de almacenamiento.

Para determinar si tu clúster tiene una actualización del sistema operativo pendiente, consultaDeterminación de las tareas de mantenimiento pendientes. Para obtener más información sobre las actualizaciones del sistema operativo, incluidos los procedimientos de la consola y la CLI, consulteActualizaciones del sistema operativo de Amazon DocumentDB.

Detención e inicio de un clúster de bases de datos de Amazon DocumentDB

La detención e inicio de los clústeres de Amazon DocumentDB puede ayudarle a administrar los costos de entornos de desarrollo y pruebas. En lugar de crear y eliminar clústeres y las instancias cada vez que utilice Amazon DocumentDB, puede detener temporalmente todas las instancias en el clúster cuando ya no son necesarias. A continuación, puede iniciarlos de nuevo cuando reanude sus pruebas.

Temas

- Información general de detención e inicio de un clúster
- Operaciones que puede llevar a cabo en un clúster detenido

Información general de detención e inicio de un clúster

Durante los periodos en los que no necesite un clúster de Amazon DocumentDB, puede detener todas las instancias de ese clúster a la vez. A continuación, puede volver a iniciar el clúster en cualquier momento que necesite usarlo. El inicio y la detención simplifican los procesos de configuración y eliminación de clústeres usados para desarrollo, pruebas o actividades similares que no requieren disponibilidad continua. Puede detener e iniciar un clúster mediante Consola de administración de AWS o AWS CLI con una sola acción, independientemente del número de instancias que haya en el clúster.

Aunque el clúster se detenga, el volumen de almacenamiento del clúster permanece sin cambios. Solo se le cobrará el almacenamiento, las instantáneas manuales y el almacenamiento de la copia de seguridad automática dentro de su intervalo de retención especificado. No se le cobrarán las horas de ninguna instancia. Después de siete días, Amazon DocumentDB vuelve a iniciar automáticamente el clúster de base de datos para asegurarse de llevar a cabo las actualizaciones de mantenimiento necesarias. Cuando el clúster comience después de siete días, se le comenzará a cobrar las instancias en el clúster de nuevo. Cuando el clúster se detenga, no podrá consultar el volumen de almacenamiento, ya que la consulta requiere que las instancias estén disponibles.

Cuando se detiene un clúster de Amazon DocumentDB, ni el clúster ni sus instancias se pueden modificar en modo alguno. Esto incluye añadir o eliminar instancias o eliminar el clúster.

Using the Consola de administración de AWS

El procedimiento siguiente le muestra cómo detener un clúster con una o varias instancias en un estado disponible o iniciar un clúster detenido.

Detención o inicio de un clúster de Amazon DocumentDB

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Clusters (Clústeres).

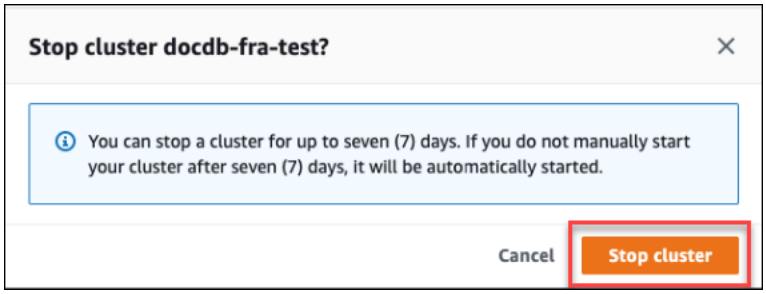
> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En la lista de clústeres, elija el botón situado a la izquierda del nombre del clúster que quiera detener o iniciar.
4. Elija Actions (Acciones) y, a continuación, elija la acción que desea realizar en el clúster.
   - Si desea detener el clúster y el clúster está disponible:
     a. Elija Detener.

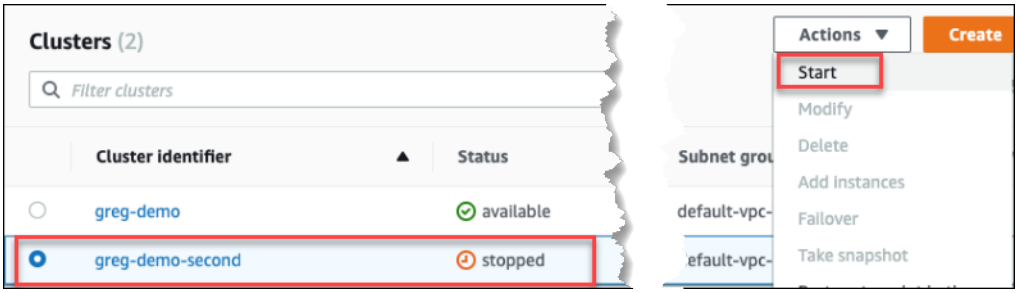


     Para evitar la activación del mecanismo de conmutación por error, la operación de detención para primero las instancias de réplica y, a continuación, la instancia principal.

     b. En el cuadro de diálogo de confirmación, acepte que desea detener el clúster seleccionado Stop cluster (Detener clúster) o mantenga el clúster en ejecución mediante Cancel (Cancelar).



   - Si desea iniciar el clúster y este se detiene, seleccione Start (Iniciar).



5. Monitorice el estado del clúster y sus instancias. Si ha iniciado el clúster, podrá volver a utilizar el clúster cuando este y sus instancias estén disponibles. Para obtener más información, consulte Determinación del estado de un clúster.



Using the AWS CLI

Los siguientes ejemplos de código le muestras cómo detener un clúster con una o varias instancias en un estado disponible o iniciar un clúster detenido.

Para detener un clúster con una o más instancias disponibles mediante AWS CLI, utilice la `stop-db-cluster` operación. Para iniciar un clúster detenido, utilice la operación `start-db-cluster`. En ambas operaciones se utiliza el parámetro `--db-cluster-identifier`.

Parámetro

- **`--db-cluster-identifier`**: obligatorio. El nombre del clúster que detener e iniciar.

Example— Para detener un clúster mediante la AWS CLI

El siguiente código detiene el clúster `sample-cluster`. El clúster debe tener una o más instancias en un estado disponible.

Para Linux, macOS o Unix:

```
aws docdb stop-db-cluster \
```

```
   --db-cluster-identifier sample-cluster
```

Para Windows:

```
aws docdb stop-db-cluster ^
   --db-cluster-identifier sample-cluster
```

Example— Para iniciar un clúster mediante el AWS CLI

El siguiente código inicia el clúster `sample-cluster`. El clúster debe estar detenido actualmente.

Para Linux, macOS o Unix:

```
aws docdb start-db-cluster \
   --db-cluster-identifier sample-cluster
```

Para Windows:

```
aws docdb start-db-cluster ^
   --db-cluster-identifier sample-cluster
```

Operaciones que puede llevar a cabo en un clúster detenido

Mientras un clúster de Amazon DocumentDB esté detenido, se podrá restaurar a cualquier momento previo que esté especificado en su intervalo de retención de copias de seguridad. Para obtener información acerca de cómo hacer una restauración en un momento dado, consulte Restauración a un momento dado.

No se puede modificar la configuración de un clúster de Amazon DocumentDB ni de ninguna de sus instancias mientras el clúster esté detenido. Tampoco puede añadir ni quitar instancias del clúster, ni eliminar el clúster si todavía tiene alguna instancia asociada. Debe iniciar el clúster antes de realizar cualquier tarea administrativa de ese tipo.

Amazon DocumentDB aplica cualquier mantenimiento programado a su clúster detenido solo después de que se vuelva a iniciar. Después de siete días, Amazon DocumentDB inicia automáticamente un clúster detenido para que no se quede demasiado rezagado en su estado de mantenimiento. Cuando el clúster se reinicie, se le comenzará a cobrar las instancias en el clúster de nuevo.

Cuando un clúster está detenido, Amazon DocumentDB no realiza los copias de seguridad automatizadas ni amplía el período de retención de copia de seguridad.

Eliminación de un clúster de Amazon DocumentDB

Puede eliminar un clúster de Amazon DocumentDB mediante Consola de administración de AWS o el. AWS CLI Para eliminar un clúster, el clúster debe tener el estado disponible y no debe tener ninguna instancia asociada a él. Si el clúster está detenido, inicie primero el clúster, espere a que el clúster esté disponible, y, a continuación, elimine el clúster. Para obtener más información, consulte Detención e inicio de un clúster de bases de datos de Amazon DocumentDB.

Protección contra eliminación

Para proteger su clúster contra la eliminación accidental, puede habilitar la protección contra eliminación. La protección contra eliminación se habilita de forma predeterminada cuando crea un clúster mediante la consola. Sin embargo, la protección contra eliminación está deshabilitada de forma predeterminada si crea un clúster con la AWS CLI.

Amazon DocumentDB aplica la protección contra eliminación para un clúster si realiza la operación de eliminación mediante la consola o la AWS CLI. Si la protección contra eliminación está habilitada, no podrá eliminar un clúster. Para eliminar un clúster con la protección contra eliminación habilitada, primero debe modificar el clúster y deshabilitar la protección contra eliminación.

Cuando use la consola con la protección contra eliminación habilitada en un clúster, no podrá eliminar la última instancia del clúster, ya que también se eliminará el clúster. Puede eliminar la última instancia de un clúster con protección contra eliminación con la AWS CLI. Sin embargo, el propio clúster sigue existiendo y se conservan los datos. Puede acceder a los datos mediante la creación de nuevas instancias para el clúster. Para obtener más información sobre cómo habilitar y deshabilitar la protección contra eliminación, consulte:

- Creación de un clúster de Amazon DocumentDB
- Modificación de un clúster de Amazon DocumentDB

Using the Consola de administración de AWS

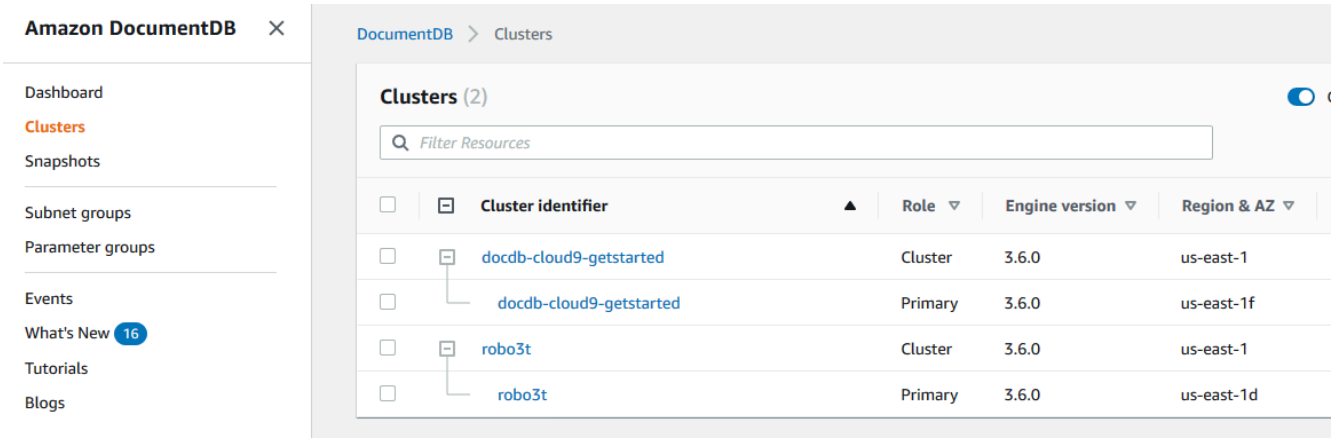
Para eliminar un clúster mediante el Consola de administración de AWS, la protección contra la eliminación debe estar deshabilitada.

Para determinar si un clúster tiene habilitada la protección contra eliminación, realice el siguiente procedimiento:

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Clusters (Clústeres).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. Tenga en cuenta que en el cuadro de navegación de clústeres, la columna Identificador de clústeres muestra tanto los clústeres como las instancias. Las instancias aparecen debajo de los clústeres, de forma similar a la siguiente captura de pantalla.



4. Elija el nombre del clúster y seleccione la pestaña Configuración. En la sección Detalles del clúster busque Protección contra eliminación. Si la protección contra eliminación está habilitada, modifique el clúster para deshabilitar la protección contra eliminación. Para obtener más información sobre la modificación de un clúster, consulte Modificación de un clúster de Amazon DocumentDB.

   Una vez que se deshabilite la protección contra eliminación, podrá eliminar el clúster.

Para eliminar un clúster:

1. En el panel de navegación, seleccione Clusters (Clústeres).

2. Determine si el clúster tiene alguna instancia; para ello, compruebe la columna Identificador de clústeres para determinar si aparecen las instancias enumeradas a continuación. Para poder eliminar un clúster, debe eliminar todas sus instancias. Para obtener más información, consulte Eliminación de una instancia de Amazon DocumentDB.
3. Si su clúster dispone de instancias, realice uno de los siguientes pasos.
   - Si el clúster no tiene instancias, seleccione el botón situado a la izquierda del nombre del clúster y elija Acciones. En el menú desplegable, seleccione Eliminar. Complete el cuadro de diálogo Delete <cluster-name> (Eliminar <nombre de clúster>) y, a continuación, elija Delete (Eliminar).
   - Si el clúster tiene una o varias instancias, haga lo siguiente:
     a. En el panel de navegación, seleccione Clusters (Clústeres).
     b. Elimine cada una de las instancias del clúster. Para ello, seleccione la casilla de verificación situada a la izquierda del nombre del clúster. Seleccione Acciones y, a continuación, elija Eliminar. Complete el cuadro de diálogo Delete <cluster-name> (Eliminar <nombre de clúster>) y, a continuación, elija Delete (Eliminar).

        Cuando elimine la última instancia, el clúster también se eliminará. Para obtener más información sobre la eliminación de instancias, consulte Eliminación de una instancia de Amazon DocumentDB.

La eliminación del clúster puede tardar varios minutos. Para monitorizar el estado del clúster, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

Using the AWS CLI

No puede eliminar un clúster que tenga instancias asociadas. Para determinar qué instancias están asociadas al clúster, ejecute el comando `describe-db-clusters` y elimine todas las instancias del clúster. A continuación, si es necesario, deshabilite la protección contra eliminación de su clúster y, por último, elimine el clúster.

1. En primer lugar, elimine todas las instancias del clúster.

   Para determinar qué instancias necesita eliminar, ejecute el siguiente comando.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
```

```
    --query 'DBClusters[*].
[DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
[
    [
        "sample-cluster",
        [
            "sample-instance-1",
            "sample-instance-2"
        ]
    ]
]
```

Si el clúster que desea eliminar tiene instancias, elimínelas como se muestra a continuación.

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-instance
```

2. En segundo lugar, deshabilite la protección contra eliminación.

   El uso de AWS CLI para eliminar todas las instancias de un clúster no elimina el clúster. También debe eliminar el clúster, pero puede hacerlo solo si se deshabilita la protección contra eliminación.

   Para determinar si el clúster tiene la protección contra eliminación habilitada, ejecute el siguiente comando.

   > **Tip**
   >
   > Para ver el estado de la protección contra eliminación de todos los clústeres de Amazon DocumentDB, omita el parámetro `--db-cluster-identifier`.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterIdentifier,DeletionProtection]'
```

La salida de esta operación será similar a lo que se indica a continuación.

```
[
    [
        "sample-cluster",
        "true"
    ]
]
```

Si el clúster tiene la protección contra eliminación habilitada, modifique el clúster y deshabilite la protección contra eliminación. Para deshabilitar la protección contra eliminación en el clúster, ejecute el siguiente comando.

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --no-deletion-protection \
    --apply-immediately
```

3. Por último, elimine el clúster.

   Una vez que la protección contra eliminación esté deshabilitada, podrá eliminar el clúster. Para eliminar un clúster, utilice la operación `delete-db-cluster` con los siguientes parámetros.

   - **`--db-cluster-identifier`**: obligatorio. El identificador del clúster que desee eliminar.
   - **`--final-db-snapshot-identifier`**: opcional. Si desea una instantánea final, debe incluir este parámetro con un nombre para la instantánea final. Debe incluir `--final-db-snapshot-identifier` o `--skip-final-snapshot`.

     Restricciones en la nomenclatura:

     - Debe tener [1-63] letras, números o guiones.
     - El primer carácter debe ser una letra.
     - No puede terminar con un guion ni contener dos guiones consecutivos.
     - Debe ser único para todos los clústeres de Amazon RDS, Amazon Neptune y Amazon DocumentDB por región Cuenta de AWS.

- **`--skip-final-snapshot`**: opcional. Use este parámetro solo si no desea realizar una instantánea final antes de eliminar el clúster. El comportamiento predeterminado es realizar una instantánea final. Debe incluir `--final-db-snapshot-identifier` o `--skip-final-snapshot`.

El siguiente AWS CLI código elimina el clúster `sample-cluster` con una instantánea final. La operación genera un error si hay las instancias asociadas al clúster o si está habilitada la protección contra eliminación.

Example

Para Linux, macOS o Unix:

```
aws docdb delete-db-cluster \
    --db-cluster-identifier sample-cluster \
    --final-db-snapshot-identifier sample-cluster-final-snapshot
```

Para Windows:

```
aws docdb delete-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --final-db-snapshot-identifier sample-cluster-final-snapshot
```

Example

El AWS CLI código siguiente elimina el clúster `sample-cluster` sin tomar una instantánea final.

Para Linux, macOS o Unix:

```
aws docdb delete-db-cluster \
    --db-cluster-identifier sample-cluster \
    --skip-final-snapshot
```

Para Windows:

```
aws docdb delete-db-cluster ^
    --db-cluster-identifier sample-cluster ^
```

```
    --skip-final-snapshot
```

El resultado de la operación `delete-db-cluster` es el clúster que se va a eliminar.

La eliminación del clúster puede tardar varios minutos. Para monitorizar el estado del clúster, consulte Monitorización del estado de un clúster.

Escalado de clústeres de Amazon DocumentDB

Amazon DocumentDB le permite escalar el almacenamiento y la computación en los clústeres en función de sus necesidades. En esta sección se describe cómo puede utilizar el escalado de almacenamiento, el escalado de instancia y el escalado de lectura para administrar el rendimiento y el escalado de los clústeres e instancias de Amazon DocumentDB.

Temas

- Escalado del almacenamiento
- Escalado de instancia
- Escalado de lectura
- Escalado de escritura

Escalado del almacenamiento

El almacenamiento de Amazon DocumentDB se escala automáticamente con los datos del volumen de clúster. A medida que aumentan los datos, el volumen de almacenamiento del clúster aumenta en incrementos de 10 GiB, hasta 256 TiB para la versión 8.0 del motor y posteriores (128 TiB para las versiones anteriores del motor).

Escalado de instancia

Puede escalar el clúster de base de datos de Amazon DocumentDB como considere necesario modificando la clase de instancia de base de datos para cada instancia de base de datos del clúster de base de datos. Amazon DocumentDB admite varias clases de instancias que están optimizadas para Amazon DocumentDB.

Para obtener más información, consulte Modificación de una instancia de base de datos de Amazon DocumentDB.

Escalado de lectura

Puede realizar el escalado de lectura de su clúster de Amazon DocumentDB creando un máximo de 15 réplicas de Amazon DocumentDB en el clúster. Cada réplica de Amazon DocumentDB devuelve los mismos datos del volumen de clúster con un retardo de réplica mínimo, normalmente menos de 100 milisegundos una vez que la instancia principal ha escrito una actualización. A medida que el tráfico de lectura aumenta, puede crear réplicas de Amazon DocumentDB adicionales y conectarlas directamente para distribuir la carga de lectura del clúster de base de datos. Las réplicas de Amazon DocumentDB no tienen que ser de la misma clase de instancia de base de datos que la instancia principal.

Para obtener más información, consulte Agregación de una instancia de Amazon DocumentDB a un clúster.

Para escalar la lectura con Amazon DocumentDB, conéctese a su clúster como un conjunto de réplicas y distribuya las lecturas a las instancias de réplica mediante las funciones de preferencias de lectura integradas en su controlador. Para obtener más información, consulte Conexión a Amazon DocumentDB como conjunto de réplicas.

Escalado de escritura

Puede escalar la capacidad de escritura en el clúster de Amazon DocumentDB aumentando el tamaño de la instancia principal del clúster. En esta sección se proporcionan dos métodos para escalar la instancia principal del clúster en función de sus necesidades. La primera opción busca minimizar el impacto de la aplicación, pero requiere completar más pasos. La segunda opción optimiza la simplicidad ya que tiene menos pasos, pero conlleva el inconveniente de tener más impacto potencial en la aplicación.

En función de la aplicación, puede elegir cuál de los enfoques siguientes es el mejor en su caso. Para obtener más información sobre los tamaños y costos de instancias disponibles, consulte la página Precios de Amazon DocumentDB.

1. Optimizar para obtener una alta disponibilidad y rendimiento: si se conecta al clúster en el modo de conjunto de réplicas (recomendado), puede utilizar el siguiente proceso para minimizar el impacto en la aplicación al escalar la instancia principal. Este método minimiza el impacto porque mantiene el clúster en alta disponibilidad o por encima de ella y los destinos de escalado de lectura se agregan al clúster como instancias, en lugar de actualizarse en su lugar.

    a. Agregue una o varias réplicas del tipo de instancia mayor al clúster (consulte ???). Asegúrese de que todas las réplicas sean del mismo tipo de instancia o de un tipo mayor

que la instancia principal. Esto evita una reducción involuntaria en el rendimiento de escritura al conmutar por error a un tipo de instancia más pequeño. En la mayoría de los casos, esto implica duplicar temporalmente la cantidad de instancias del clúster y, a continuación, eliminar las réplicas más pequeñas una vez finalizado el escalado.

b. Establezca el nivel de conmutación por error en todas las réplicas nuevas en prioridad cero, garantizando que una réplica del tipo de instancia más pequeño tenga la prioridad de conmutación por error más alta. Para obtener más información, consulte ???.

c. Inicie una conmutación por error manual, que promoverá una de las nuevas réplicas para que sea la instancia principal. Para obtener más información, consulte ???.

> **Note**
>
> Esto incurrirá en ~30 segundos de tiempo de inactividad para el clúster. Planifíquelo en consecuencia.

d. Elimine todas las réplicas de un tipo de instancia menor que la nueva instancia principal del clúster.

e. Vuelva a establecer el nivel de conmutación por error de todas las instancias en la misma prioridad (normalmente, esto significa volver a establecerlas en 1).

Como ejemplo, suponga que tiene un clúster que contiene actualmente tres instancias `r5.large` (una principal y dos réplicas) y desea escalar a un tipo de instancia `r5.xlarge`. Para ello, primero agregaría tres instancias de réplica `r5.xlarge` al clúster y, a continuación, establecería el nivel de conmutación por error de las nuevas réplicas `r5.xlarge` en cero. A continuación, iniciaría una conmutación por error manual (entendiendo que su aplicación experimentará ~30 segundos de tiempo de inactividad). Una vez completada la conmutación por error, eliminaría las tres instancias `r5.large` del clúster, dejando el clúster escalado a instancias `r5.xlarge`.

Para ayudar a optimizar los costos, las instancias de Amazon DocumentDB se facturan en incrementos de un segundo con un cargo mínimo de 10 minutos a partir de la implementación de un cambio de estado que se pueda facturar, como la creación, la modificación o la eliminación de una instancia. Para obtener más información, consulte Optimización de costos en la documentación de prácticas recomendadas.

2. Optimizar la simplicidad: este enfoque optimiza la simplicidad. No expande ni contrae el clúster, pero podría reducir temporalmente la capacidad de lectura.

Es posible que al cambiar la clase de instancia de una réplica, la instancia deje de atender las solicitudes durante un breve período de tiempo, desde unos pocos segundos a menos de 30 segundos. Si se conecta al clúster en el modo de conjunto de réplicas (recomendado), esto reduciría su capacidad de lectura en una réplica (por ejemplo, hasta el 66 % de la capacidad en un clúster de 3 nodos o el 75 % de la capacidad en un clúster de 4 nodos, etc.) durante la operación de escalado.

a. Escale una de las instancias de réplica del clúster. Para obtener más información, consulte Administración de clases de instancias.

b. Espere hasta que la instancia esté disponible (consulte Supervisión del estado de un clúster de Amazon DocumentDB).

> **Note**
>
> Esto incurrirá en ~30 segundos de tiempo de inactividad para el clúster. Planifíquelo en consecuencia.

c. Continúe ejecutando los pasos 1 y 2 hasta que se hayan escalado todas las instancias de réplicas, una por una.

d. Iniciar una conmutación por error manual. Esto promoverá una de las réplicas para que sea la instancia principal. Para obtener más información, consulte Conmutación por error de Amazon DocumentDB.

> **Note**
>
> Esto provocará un tiempo de inactividad de hasta 30 segundos para el clúster, aunque suele llevar menos tiempo. Planifíquelo en consecuencia.

e. Escale la anterior instancia principal (que ahora es una réplica).

Clonación de un volumen de clúster de base de datos de Amazon DocumentDB

Con la clonación de Amazon DocumentDB, puede crear un nuevo clúster que utilice el mismo volumen de clúster de Amazon DocumentDB y tenga los mismos datos que el original. El proceso está diseñado para ser rápido y rentable. El nuevo clúster con su volumen de datos asociado se

conoce como clon. La creación de un clon es más rápido y más eficiente en el espacio que copiar físicamente los datos mediante una técnica diferente, como la restauración de una instantánea.

Amazon DocumentDB admite la creación de un clon aprovisionado de Amazon DocumentDB a partir de un clúster de Amazon DocumentDB aprovisionado. Cuando crea un clon con una configuración de implementación diferente a la de origen, el clon se crea con la versión secundaria más reciente del motor de base de datos Amazon DocumentDB.

Cuando crea clones a partir de sus clústeres de Amazon DocumentDB, los clones se crean en su cuenta, la misma cuenta que es propietaria del AWS clúster de Amazon DocumentDB de origen.

Temas

- Información general de la clonación de Amazon DocumentDB
- Notas de uso para la clonación de Amazon DocumentDB
- Cómo funciona la clonación de Amazon DocumentDB
- Creación de un clon de Amazon DocumentDB

Información general de la clonación de Amazon DocumentDB

Amazon DocumentDB utiliza un protocolo de copia en escritura para crear un clon. Este mecanismo utiliza un espacio adicional mínimo para crear un clon inicial. Cuando se crea el clon por primera vez, Amazon DocumentDB guarda una sola copia de los datos que utiliza el clúster de base de datos de Amazon DocumentDB de origen y el nuevo clúster de base de datos de Amazon DocumentDB (clonado). El almacenamiento adicional solo se asigna cuando el clúster de base de datos de Amazon DocumentDB de origen o el clon del clúster de base de datos de Amazon DocumentDB realizan cambios en los datos (en el volumen de almacenamiento de Amazon DocumentDB). Para obtener más información sobre el protocolo de copia en escritura, consulte Cómo funciona la clonación de Amazon DocumentDB.

La clonación de Amazon DocumentDB es especialmente útil para configurar rápidamente entornos de prueba mediante sus datos de producción, sin riesgo de corrupción de datos. Puede utilizar clones para muchos tipos de aplicaciones de corta duración, como las siguientes:

- Experimente con cambios potenciales (por ejemplo, cambios de esquema y cambios de grupo de parámetros) para evaluar todos los impactos.
- Realice operaciones intensivas de carga de trabajo, como exportar datos o ejecutar consultas analíticas en el clon.

- Cree una copia del clúster de base de datos de producción para desarrollo, pruebas u otros fines.

Puede crear más de un clon desde el mismo clúster de base de datos de Amazon DocumentDB. También puede crear varios clones desde otro clon.

Tras crear un clon de Amazon DocumentDB, puede configurar las instancias de Amazon DocumentDB de forma diferente a la del clúster de Amazon DocumentDB de origen. Por ejemplo, es posible que no necesite un clon con fines de desarrollo para cumplir con los mismos requisitos de alta disponibilidad que el clúster de base de datos Amazon DocumentDB de producción de origen. En este caso, puede configurar el clon con una única instancia de base de datos de Amazon DocumentDB en lugar de las múltiples instancias de base de datos utilizadas por el clúster de base de datos de Amazon DocumentDB.

Cuando termine de utilizar el clon para realizar pruebas, desarrollo u otros fines, puede eliminarlo.

Notas de uso para la clonación de Amazon DocumentDB

La clonación de Amazon DocumentDB tiene las siguientes limitaciones:

- Puede crear tantos clones como desee, hasta el número máximo de clústeres de base de datos permitido en la Región de AWS. No obstante, después de crear 15 clones, el siguiente es una copia completa. La operación de clonación actúa como una recuperación a un momento dado.
- No puede crear un clon en una AWS región diferente del clúster de Amazon DocumentDB de origen.
- No se puede crear un clon desde un clúster de base de datos de Amazon DocumentDB que no tiene instancias de base de datos. Solo se pueden clonar clústeres de base de datos de Amazon DocumentDB que tengan al menos una instancia de base de datos.
- Se puede crear un clon en una nube privada virtual (VPC) diferente de la del clúster de base de datos de Amazon DocumentDB. Sin embargo, las subredes de esas VPC deben estar asignadas al mismo conjunto de zonas de disponibilidad.

Cómo funciona la clonación de Amazon DocumentDB

La clonación de Amazon DocumentDB funciona en la capa de almacenamiento de un clúster de Amazon DocumentDB. Utiliza un protocolo copy-on-write que es rápido y eficiente en el espacio en términos de los medios permanentes subyacentes que soportan el volumen de almacenamiento de Amazon DocumentDB. Puede obtener más información sobre volúmenes de clúster de Amazon DocumentDB en Administración de clústeres de Amazon DocumentDB.

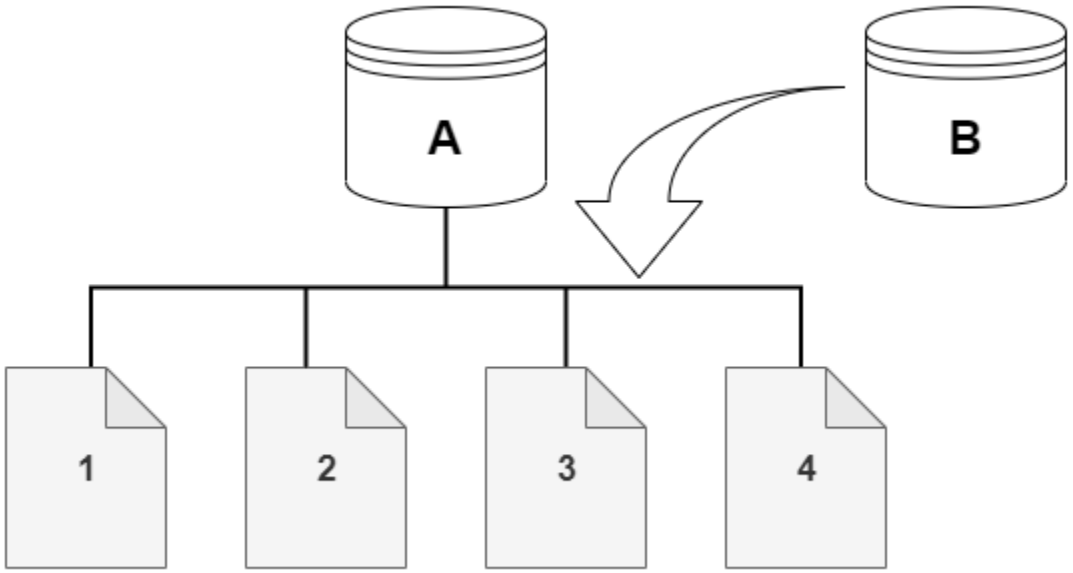
Temas

- Descripción del protocolo de copia en escritura
- Eliminación de un volumen del clúster de origen

Descripción del protocolo de copia en escritura

Un clúster de Amazon DocumentDB almacena datos en páginas en el volumen de almacenamiento de Amazon DocumentDB subyacente.

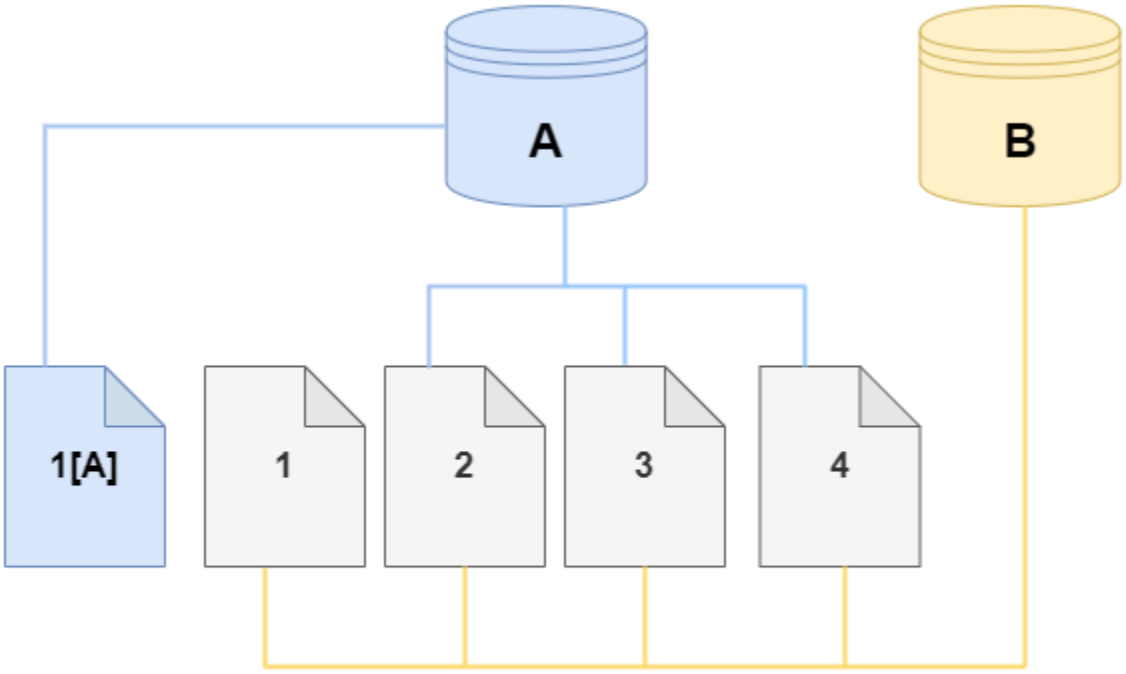
Por ejemplo, en el siguiente diagrama puede encontrar un clúster de base de datos de Amazon DocumentDB (A) que tiene cuatro páginas de datos, 1, 2, 3 y 4. Imagine que un clon, B, se crea desde del clúster de base de datos de Amazon DocumentDB. Cuando se crea el clon, no se copian datos. Más bien, el clon apunta al mismo conjunto de páginas que el clúster de base de datos de Amazon DocumentDB de origen.



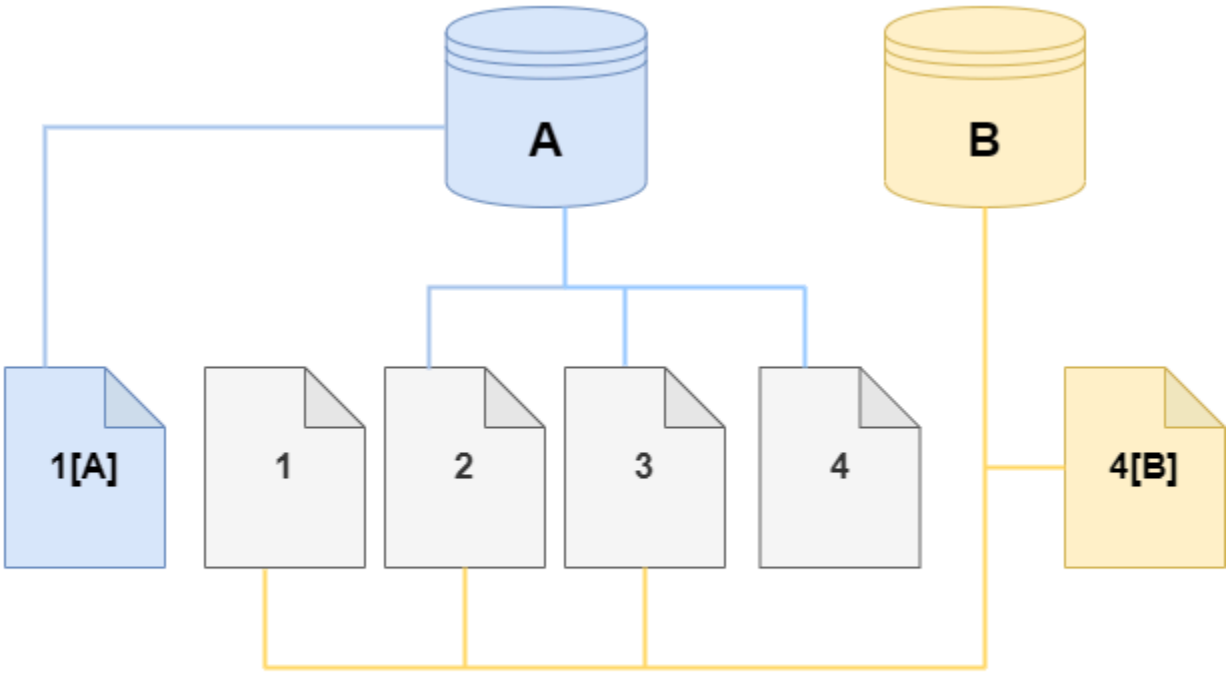
Cuando se crea el clon, generalmente no se necesita almacenamiento adicional. El protocolo de copia en escritura utiliza el mismo segmento en los medios de almacenamiento físico que el segmento de origen. Solo se requiere almacenamiento adicional si la capacidad del segmento de origen no es suficiente para todo el segmento de clones. Si ese es el caso, el segmento de origen se copia en otro dispositivo físico.

En los diagramas siguientes, puede encontrar un ejemplo del protocolo de copia en escritura en acción utilizando el mismo clúster A y su clon B, como se muestra anteriormente. Supongamos

que realiza un cambio en su clúster de base de datos de Amazon DocumentDB (A) que da lugar a un cambio en los datos almacenados en la página 1. En lugar de escribir en la página original 1, Amazon DocumentDB crea una nueva página 1 [A]. El volumen del clúster de base de datos de Amazon DocumentDB para el clúster (A) ahora apunta a la página 1 [A], 2, 3 y 4, mientras que el clon (B) sigue haciendo referencia a las páginas originales.



En el clon, se realiza un cambio en la página 4 del volumen de almacenamiento. En lugar de escribir en la página original 4, Amazon DocumentDB crea una nueva página 4 [B]. El clon ahora apunta a las páginas 1, 2, 3 y a la página 4 [B], mientras que el clúster (A) continúa apuntando a 1 [A], 2, 3 y 4.



A medida que se producen más cambios a lo largo del tiempo en el clon y el volumen del clúster de base de datos de Amazon DocumentDB de origen, necesitará cada vez más almacenamiento para capturar y almacenar los cambios.

Eliminación de un volumen del clúster de origen

Cuando elimina un volumen del clúster de origen que tiene uno o más clones asociados, los clones no se ven afectados. Los clones siguen apuntando a las páginas que antes pertenecían al volumen del clúster de origen.

Creación de un clon de Amazon DocumentDB

Puede crear un clon en la misma AWS cuenta que el clúster de Amazon DocumentDB de origen. Para ello, puede utilizar los procedimientos siguientes Consola de administración de AWS o AWS CLI los siguientes.

Al utilizar la clonación de Amazon DocumentDB, puede crear un clon de clúster de Amazon DocumentDB aprovisionado a partir de un clúster de Amazon DocumentDB aprovisionado.

Using the Consola de administración de AWS

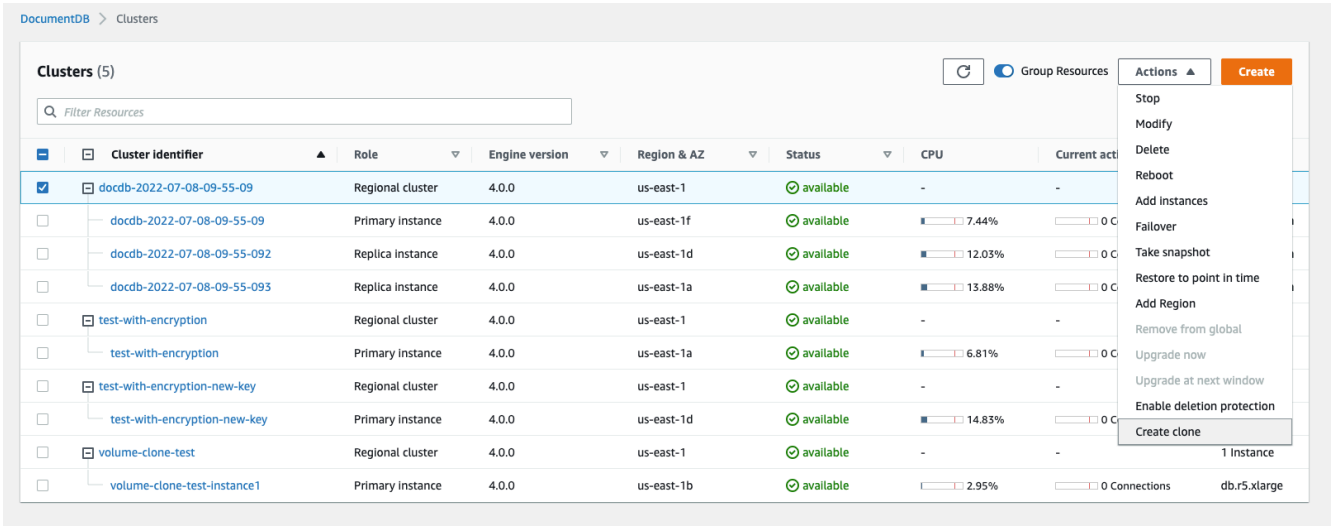
El siguiente procedimiento describe cómo clonar un clúster de Amazon DocumentDB usando la Consola de administración de AWS.

Crear un clon con los Consola de administración de AWS resultados en un clúster de Amazon DocumentDB con una instancia de Amazon DocumentDB.

Estas instrucciones se aplican a los clústeres de bases de datos que pertenecen a la misma AWS cuenta que está creando el clon. El clúster de base de datos debe pertenecer a la misma AWS cuenta, ya que Amazon DocumentDB no admite la clonación entre cuentas.

Para crear un clon de un clúster de base de datos que sea propiedad de su AWS cuenta que utiliza el Consola de administración de AWS

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Clusters (Clústeres).
3. Elija su clúster de base de datos de Amazon DocumentDB de la lista y para Acciones, elija Crear clon.

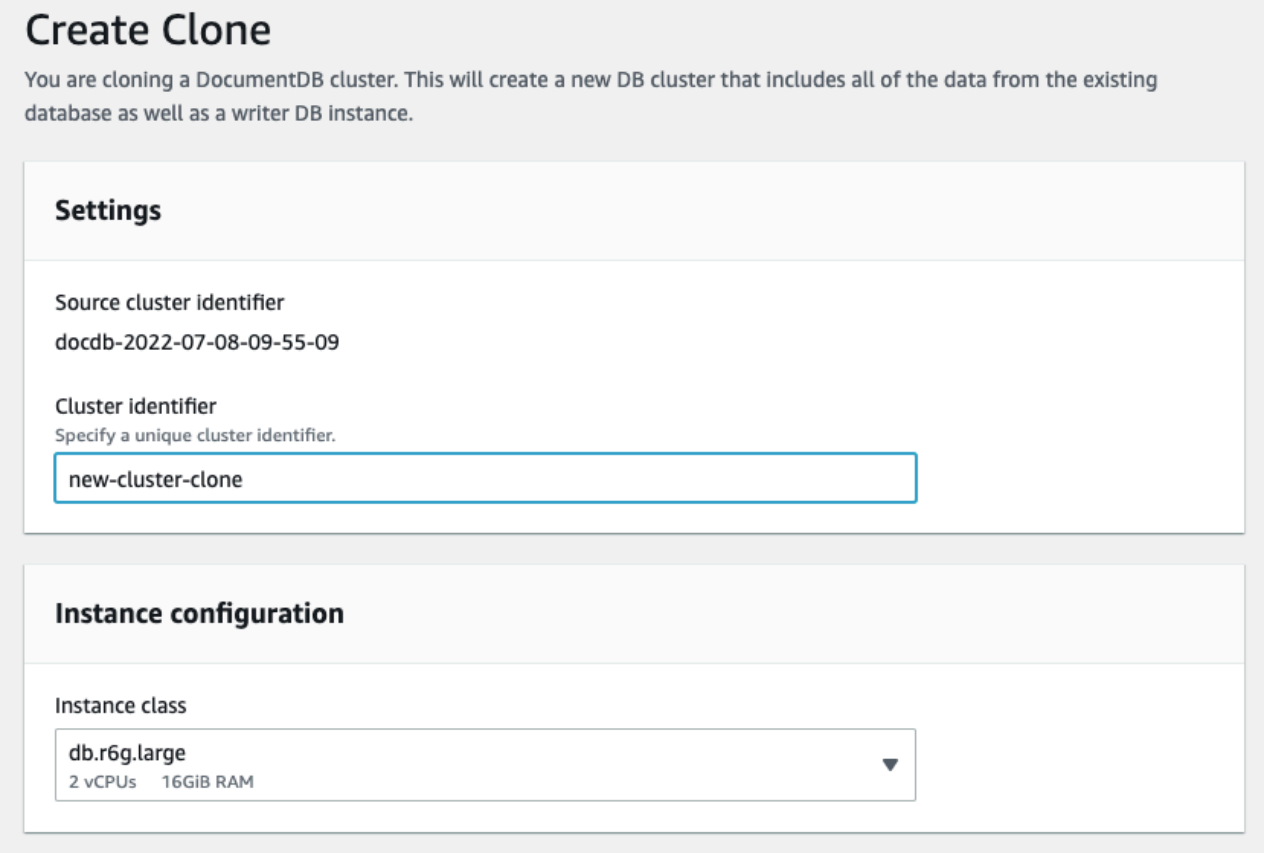


Se abre la página Crear clon, donde puede configurar un Identificador de clúster y una Clase de instancia, y otras opciones para el clon del clúster de Amazon DocumentDB.

4. En la sección Settings, realice lo siguiente:

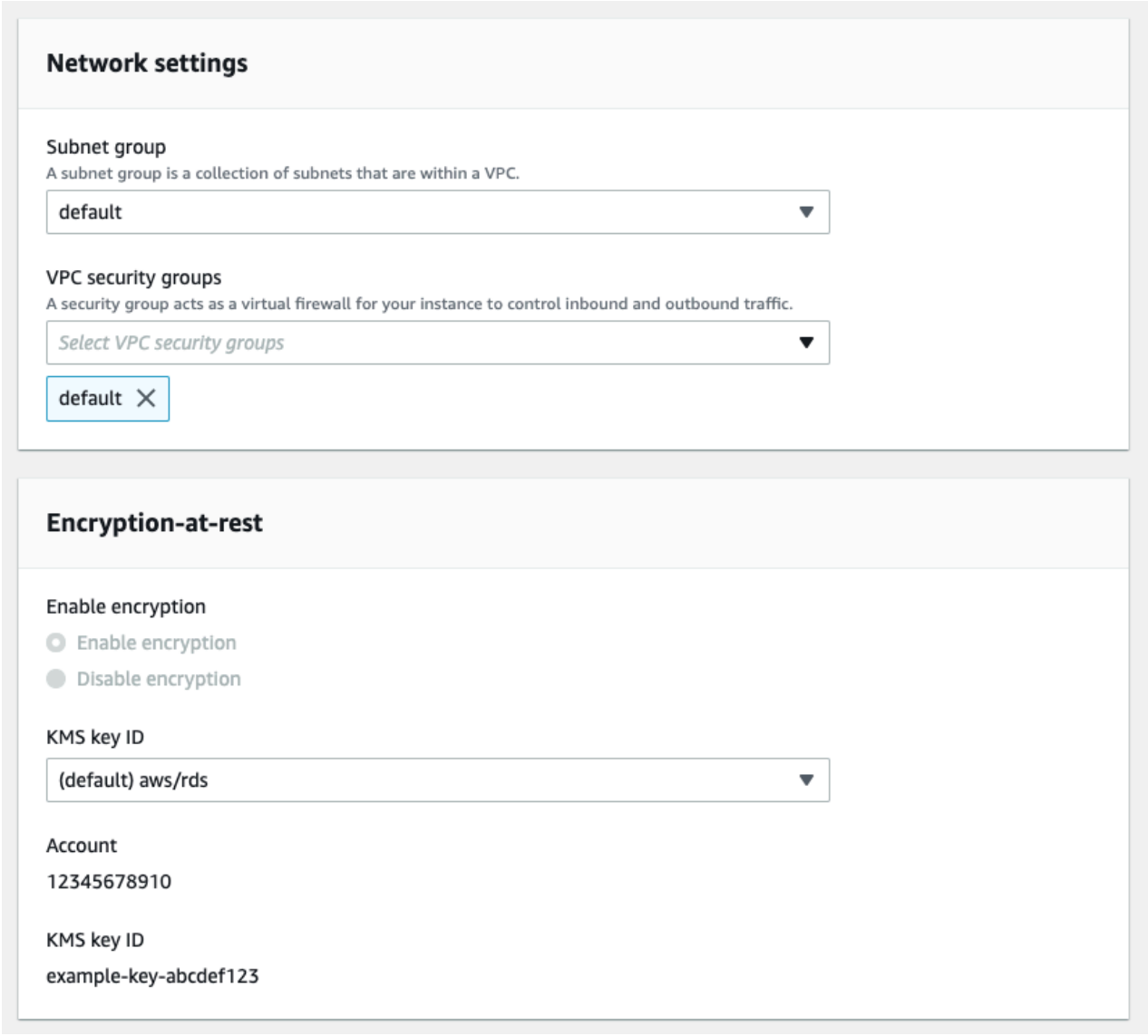
   a. Para el identificador de clúster de base de datos, ingrese el nombre que desea dar a su clúster de base de datos de Amazon DocumentDB clonado.

b. Para la Configuración de instancias, seleccione una Clase de instancia adecuada para su clúster clonado de Amazon DocumentDB.

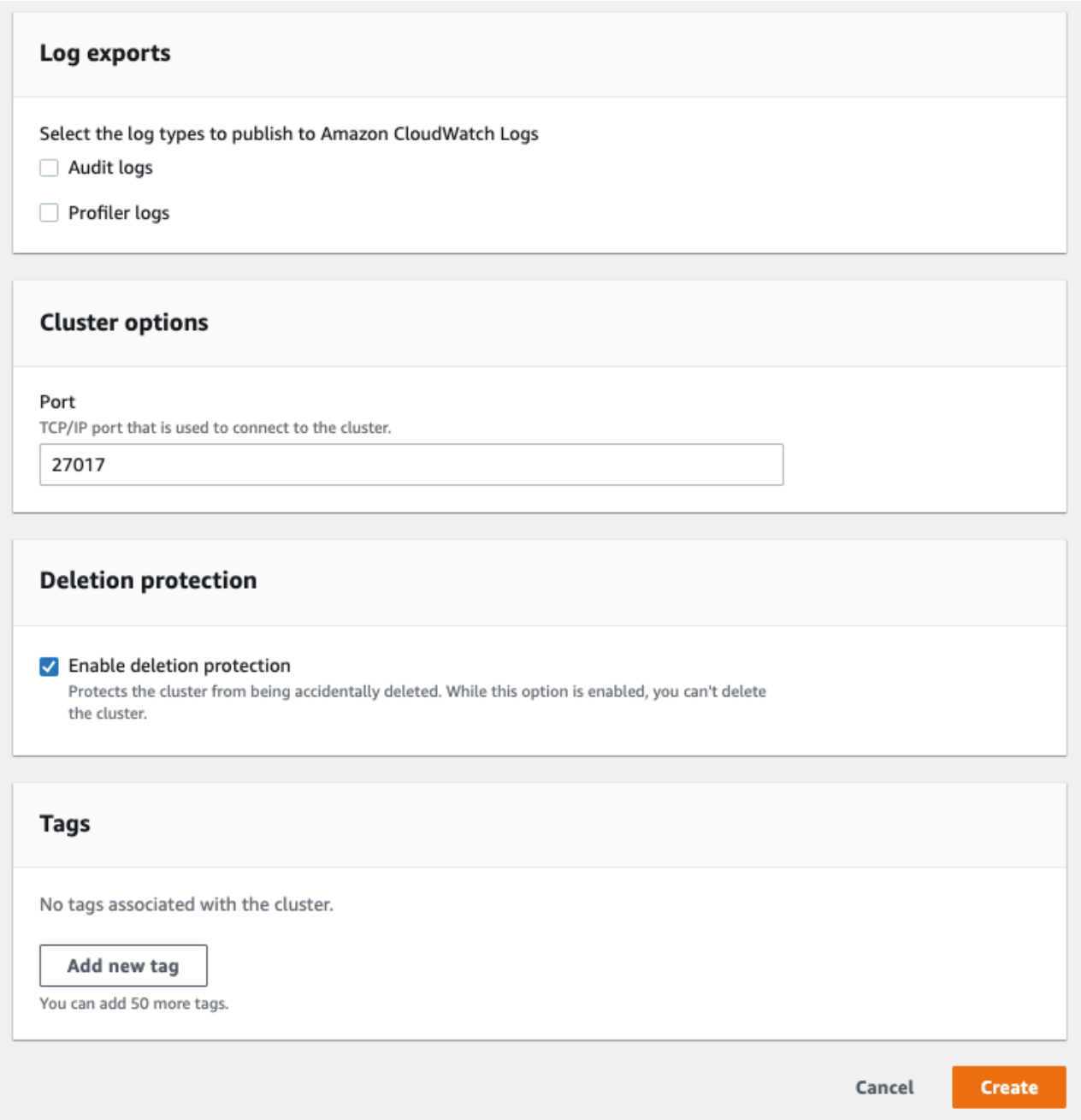


c. En la Configuración de red, elija un Grupo de subredes para su caso de uso y los grupos de seguridad de VPC asociados.

d. Pues Encryption-at-rest, si el clúster de origen (el clúster que se está clonando) tiene el cifrado habilitado, el clúster clonado también debe tener el cifrado habilitado. Si se da este escenario, las opciones de Habilitar el cifrado aparecen atenuadas (deshabilitadas), pero con la opción Habilitar el cifrado seleccionada. Por el contrario, si el clúster de origen no tiene el cifrado habilitado, están disponibles las opciones de Habilitar el cifrado y puede elegir habilitar o deshabilitar el cifrado.



e. Complete la nueva configuración del clon del clúster seleccionando el tipo de registros que desea exportar (opcional), introduciendo un puerto específico utilizado para conectarse al clúster y habilitando la protección contra la eliminación accidental del clúster (esta opción está habilitada de forma predeterminada).



f. Termine de ingresar todos los ajustes para su clon de clúster de base de datos de Amazon DocumentDB. Para obtener más información sobre la configuración del clúster y de la instancia de base de datos de Amazon DocumentDB, consulte Administración de clústeres de Amazon DocumentDB.

5. Seleccione Crear clon para lanzar el clon de Amazon DocumentDB del clúster de Amazon DocumentDB que haya elegido.

Cuando se crea el clon, aparece junto con los otros clústeres de base de datos de Amazon DocumentDB en la sección Bases de datos de la consola y muestra su estado actual. Su clon está listo para utilizar cuando su estado es Available (Disponible).

Using the AWS CLI

El uso del AWS CLI para clonar su clúster de Amazon DocumentDB implica un par de pasos.

El `restore-db-cluster-to-point-in-time` AWS CLI comando que utiliza da como resultado un clúster de Amazon DocumentDB vacío con 0 instancias de Amazon DocumentDB. Es decir, el comando restaura solo el clúster de base de datos de Amazon DocumentDB, no las instancias de base de datos de dicho clúster. Lo hace por separado después de que el clon está disponible. Los dos pasos en el proceso son los siguientes:

1. Cree el clon mediante el comando restore-db-cluster-to-point-in-time de la CLI. Los parámetros que utiliza con este comando controlan el tipo de capacidad y otros detalles del clúster vacío (clon) de base de datos de Amazon DocumentDB que se está creando.
2. Cree la instancia de base de datos de Amazon DocumentDB para el clon mediante el comando create-db-instance de la CLI para volver a crear la instancia de base de datos de Amazon DocumentDB en el clúster de base de datos de Amazon DocumentDB restaurado.

Los siguientes comandos asumen que AWS CLI está configurado con su AWS región como predeterminada. Este enfoque le ahorra pasar el nombre de `--region` en cada uno de los comandos. Para obtener más información, consulte Configuración de la AWS CLI. También puede especificar la `--region` en cada uno de los comandos de la CLI que siguen.

Creación del clon

Los parámetros específicos que se pasan al comando `restore-db-cluster-to-point-in-time` de la CLI varían. Lo que pase dependerá del tipo de clon que desea crear.

Utilice el siguiente procedimiento para crear un clon de Amazon DocumentDB aprovisionado a partir de un clúster de Amazon DocumentDB aprovisionado.

Creación de un clon del mismo modo de motor que el clúster de base de datos de Amazon DocumentDB de origen

- Utilice el comando `restore-db-cluster-to-point-in-time` de la CLI y especifique los valores para los siguientes parámetros:

- `--db-cluster-identifier`: elija un nombre significativo para su clon. Se asigna un nombre al clon cuando se utiliza el comando restore-db-clúster-to-point-in-time de la CLI.
- `--restore-type`: utilice `copy-on-write` para crear un clon del clúster de base de datos de origen. Sin este parámetro, `restore-db-cluster-to-point-in-time` restaura el clúster de base de datos de Amazon DocumentDB en lugar de crear un clon. El valor predeterminado para `restore-type` es `full-copy`.
- `--source-db-cluster-identifier`: utilice el nombre del clúster de base de datos de Amazon DocumentDB de origen que desea clonar.
- `--use-latest-restorable-time`: este valor apunta a los datos de volumen restaurables más recientes para el clon. Este parámetro es obligatorio para `restore-type copy-on-write`, pero no se puede utilizar `restore-to-time parameter` con él.

El siguiente ejemplo crea un clon del clúster de denominado `my-clone` desde un clúster denominado `my-source-cluster`.

Para 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
```

Para 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
```

El comando devuelve el objeto JSON que contiene detalles del clon. Compruebe que su clúster de base de datos clonado está disponible antes de intentar crear la instancia de base de datos para su clon. Para obtener más información, consulte Comprobar el estado y obtener detalles del clon a continuación:

Comprobación del estado y obtención de detalles del clon

Puede utilizar el siguiente comando para verificar el estado del clúster de base de datos vacío recién creado.

```
$ aws docdb describe-db-clusters --db-cluster-identifier my-clone --query '*[].
[Status]' --output text
```

O bien, puede obtener el estado y los demás valores que necesita para crear la instancia de base de datos para su clon mediante la siguiente AWS CLI consulta:

Para Linux, macOS o Unix:

```
aws docdb describe-db-clusters --db-cluster-identifier my-clone \
  --query '*[].{Status:Status,Engine:Engine,EngineVersion:EngineVersion}'
```

Para Windows:

```
aws docdb describe-db-clusters --db-cluster-identifier my-clone ^
  --query "*[].{Status:Status,Engine:Engine,EngineVersion:EngineVersion}"
```

Esta consulta devuelve un resultado similar al siguiente:

```
[
  {
        "Status": "available",
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
    }
]
```

Creación de la instancia de Amazon DocumentDB para su clon

Llame al comando create-db-instance de la CLI para crear la instancia de base de datos para su clon.

El parámetro `--db-instance-class` se utiliza sólo para clústeres de base de datos aprovisionados de Amazon DocumentDB.

Para 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
```

Para 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
```

Parámetros para utilizar durante la clonación

En la siguiente tabla se resumen los diversos parámetros utilizados con `restore-db-cluster-to-point-in-time` para clonar clústeres de base de datos de Amazon DocumentDB.

Parámetro	Descripción
--source-db-cluster-identifier	Utilice el nombre del clúster de base de datos de Amazon DocumentDB de origen que desea clonar.
--db-cluster-identifier	Elija un nombre significativo para su clon. Asigne un nombre a su clon con el comando `restore-db-cluster-to-point-in-time`. A continuación, pase este nombre al comando `create-db-instance` .
—restore-type	Especifique `copy-on-write` como el `--restore-type` para crear un clon del clúster de base de datos de origen en lugar de restaurar el clúster de base de datos de Amazon DocumentDB de origen.
--use-latest-restorable-time	Este valor apunta a los datos de volumen restaurables más recientes para el clon.

Información general de la tolerancia a errores de clúster de Amazon DocumentDB

Los clústeres de Amazon DocumentDB ofrecen tolerancia a errores por diseño. El volumen de cada clúster abarca varias zonas de disponibilidad en una sola Región de AWS, y cada zona de disponibilidad contiene una copia de los datos del volumen del clúster. Esta funcionalidad significa que el clúster puede tolerar un error de una zona de disponibilidad sin perder datos y con tan solo una interrupción breve del servicio.

Si se produce un error en la instancia principal de un clúster de base de datos, Amazon DocumentDB realiza automáticamente una conmutación por error a una nueva instancia principal de una de las dos formas siguientes:

- Promocionando una réplica de Amazon DocumentDB existente a la nueva instancia principal elegida en función de la configuración del nivel de promoción de cada réplica y, a continuación, creando una que sustituya a la anterior principal. Una conmutación por error a la instancia de réplica suele tardar menos de 30 segundos. Las operaciones de lectura y escritura pueden sufrir breves interrupciones durante este período. Para aumentar la disponibilidad del clúster, cree al menos una o más réplicas de Amazon DocumentDB en dos o más zonas de disponibilidad diferentes.
- Creando una nueva instancia principal. Esto solo ocurre si no tiene una instancia de réplica en el clúster y puede tardar unos minutos en completarse.

Si el clúster tiene una o varias réplicas de Amazon DocumentDB, se promueve una réplica de Amazon DocumentDB a instancia principal durante un evento de error. Un evento de error provoca una interrupción breve durante la cual las operaciones de lectura y escritura generan errores con una excepción. Sin embargo, el servicio se suele restaurar en menos de 120 segundos y, en muchos casos, en menos de 60 segundos. Para aumentar la disponibilidad de su clúster, cree al menos una o más réplicas de Amazon DocumentDB en dos o más zonas de disponibilidad diferentes.

Puede personalizar el orden en que se promueven las réplicas de Amazon DocumentDB a instancia principal tras un error mediante la asignación de una prioridad a cada réplica. Las prioridades van desde 0 para la prioridad más alta hasta 15 para la más baja. Si la instancia principal experimenta un error, la réplica de Amazon DocumentDB que tenga la máxima prioridad pasará a ser la instancia principal. Puede modificar la prioridad de una réplica de Amazon DocumentDB en cualquier momento. Al modificar la prioridad, no se activa una conmutación por error. Puede usar la operación `modify-db-instance` con el parámetro `--promotion-tier`. Para obtener más información

acerca de cómo personalizar la prioridad de conmutación por error de una instancia, consulte Conmutación por error de Amazon DocumentDB.

Puede haber más de una réplica de Amazon DocumentDB con la misma prioridad, lo que genera niveles de promoción. Si dos o más réplicas de Amazon DocumentDB comparten la misma prioridad, pasará a ser la instancia principal la réplica que tiene un tamaño mayor. Si dos o más réplicas de Amazon DocumentDB tienen la misma prioridad y el mismo tamaño, se promueve una réplica arbitraria del mismo nivel de promoción.

Si el clúster no contiene ninguna réplica de Amazon DocumentDB, la instancia principal se vuelve a crear durante un evento de error. Un evento de error provoca una interrupción durante la cual las operaciones de lectura y escritura generan errores con una excepción. El servicio se restaura cuando se crea la nueva instancia principal, un proceso que normalmente dura menos de 10 minutos. Promover una réplica de Amazon DocumentDB a instancia principal es mucho más rápido que crear una nueva instancia principal.

Administración de instancias de Amazon DocumentDB

Los siguientes temas proporcionan información que le ayudarán a administrar sus instancias de Amazon DocumentDB. Incluyen detalles sobre las clases y los estados de instancias, y cómo crear, eliminar y modificar una instancia.

Temas

- Determinación del estado de una instancia
- Ciclo de vida de instancia de Amazon DocumentDB
- Administración de clases de instancias
- NVMe-backed instancias

Determinación del estado de una instancia

Para ver los estados válidos de las instancias, su significado y saber cómo determinar el estado de las instancias, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

Ciclo de vida de instancia de Amazon DocumentDB

El ciclo de vida de una instancia Amazon DocumentDB comprende su creación, modificación, mantenimiento y actualización, ejecución de copias de seguridad y restauraciones, reinicio,

actualización y eliminación. En esta sección se proporciona información sobre cómo realizar estos procesos.

Temas

- Agregación de una instancia de Amazon DocumentDB a un clúster
- Descripción de las instancias de Amazon DocumentDB
- Modificación de una instancia de base de datos de Amazon DocumentDB
- Reinicio de la instancia de Amazon DocumentDB
- Eliminación de una instancia de Amazon DocumentDB

Puede crear una nueva instancia de Amazon DocumentDB mediante el Consola de administración de AWS o el. AWS CLI Para añadir una instancia a un clúster, el clúster debe tener un estado disponible. No puede añadir una instancia a un clúster que está detenido. Si el clúster está detenido, inicie primero el clúster, espere a que el clúster esté disponible y, a continuación, añada una instancia. Para obtener más información, consulte Detención e inicio de un clúster de bases de datos de Amazon DocumentDB.

> **Note**
>
> Si crea un clúster de Amazon DocumentDB mediante la consola, se crea automáticamente una instancia al mismo tiempo. Si desea crear instancias adicionales, utilice uno de los siguientes procedimientos.

Agregación de una instancia de Amazon DocumentDB a un clúster

Using the Consola de administración de AWS

El siguiente procedimiento permite crear una instancia de su clúster mediante la consola de Amazon DocumentDB.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Clusters (Clústeres).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. Para elegir el clúster al que desea añadir una instancia, seleccione el botón situado a la izquierda del nombre del clúster.
4. Elija Actions (Acciones) y después Add instances (Añadir instancias).
5. En la página Add instance to (Añadir instancia a)<cluster-name>, repita los pasos que se describen a continuación para cada instancia que desee añadir al clúster. Puede tener un máximo de 15.

   a. Instance identifier: puede escribir un identificador único para esta instancia o permitir que Amazon DocumentDB proporcione el identificador de la instancia basándose en el identificador del clúster.

      Restricciones en cuanto a la nomenclatura de las instancias:

      - Debe tener [1-63] letras, números o guiones.
      - El primer carácter debe ser una letra.
      - No puede terminar con un guion ni contener dos guiones consecutivos.
      - Debe ser único para todas las instancias de Amazon RDS, Neptune y Amazon DocumentDB por región. Cuenta de AWS

   b. Clase de instancia: en la lista desplegable, elija el tipo de instancia que desea para esta instancia.

   c. Capa de promoción: en la lista, elija la capa de promoción de la instancia, o bien elija Sin preferencia para permitir que Amazon DocumentDB establezca la capa de promoción de la instancia. Un número menor indica una mayor prioridad. Para obtener más información, consulte Control del destino de la conmutación por error.

   d. Para añadir más instancias, elija Add additional instances (Agregar instancias adicionales) y repita los pasos a, b y c.

6. Finalizar la operación.

   - Para añadir las instancias a su clúster, seleccione Create (Crear).

- Para cancelar la operación, elija Cancel (Cancelar).

La creación de una instancia puede tardar varios minutos. Puede usar la consola o ver el AWS CLI estado de la instancia. Para obtener más información, consulte Monitorización del estado de una instancia.

Using the AWS CLI

Usa la `create-db-instance` AWS CLI operación con los siguientes parámetros para crear la instancia principal del clúster.

- **`--db-instance-class`**: obligatorio. La capacidad de memoria e informática de la instancia (por ejemplo, `db.m4.large`). No todas las clases de instancia están disponibles en todas las Regiones de AWS.
- **`--db-instance-identifier`**: obligatorio. Una cadena que identifica la instancia.

  Restricciones en cuanto a la nomenclatura de las instancias:

  - Debe tener [1-63] letras, números o guiones.
  - El primer carácter debe ser una letra.
  - No puede terminar con un guion ni contener dos guiones consecutivos.
  - Debe ser único para todas las instancias de Amazon RDS, Neptune y Amazon DocumentDB por región. Cuenta de AWS
- **`--engine`**: obligatorio. Debe ser `docdb`.
- **`--availability-zone`**: opcional. La zona de disponibilidad en la que desea crear esta instancia. Use este parámetro para colocar sus instancias en diferentes zonas de disponibilidad con el fin de aumentar la tolerancia a errores. Para obtener más información, consulte Alta disponibilidad y replicación de Amazon DocumentDB.
- **`--promotion-tier`**: opcional. El nivel de prioridad de conmutación por error de esta instancia. Debe ser un número comprendido entre 0 y 15; un número menor indica una mayor prioridad. Para obtener más información, consulte Control del destino de la conmutación por error.

1. En primer lugar, determine en qué zonas de disponibilidad puede crear la instancia.

Si desea especificar la zona de disponibilidad, antes de crear la instancia, ejecute el siguiente comando para determinar en qué zonas de disponibilidad están disponibles para el clúster de Amazon DocumentDB.

Para Linux, macOS o Unix:

```
aws docdb describe-db-clusters \
       --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
```

Para Windows:

```
aws docdb describe-db-clusters ^
       --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
```

La salida de esta operación será similar a lo que se indica a continuación.

```
[
    [
        "sample-cluster",
        [
            "us-east-1c",
            "us-east-1b",
            "us-east-1a"
        ]
    ]
]
```

2. En segundo lugar, determine qué clases de instancias puede crear en su región.

   Para determinar qué clases de instancias tiene a su disposición en su región, ejecute el siguiente comando. En el resultado, elija una clase de instancia para la instancia que desea agregar al clúster de Amazon DocumentDB.

   Para Linux, macOS o Unix:

```
aws docdb describe-orderable-db-instance-options \
        --engine docdb \
        --query 'OrderableDBInstanceOptions[*].DBInstanceClass'
```

Para Windows:

```
aws docdb describe-orderable-db-instance-options ^
        --engine docdb ^
        --query 'OrderableDBInstanceOptions[*].DBInstanceClass'
```

La salida de esta operación será similar a lo que se indica a continuación.

```
[
    "db.r5.16xlarge",
    "db.r5.2xlarge",
    "db.r5.4xlarge",
    "db.r5.8xlarge",
    "db.r5.large",
    "db.r5.xlarge"
]
```

3. Agregar una instancia a su clúster de Amazon DocumentDB.

   Para agregar una instancia al clúster de Amazon DocumentDB, ejecute el siguiente comando.

   Para 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
```

   Para 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
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "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 creación de la instancia puede tardar varios minutos. Puede usar la consola o ver el AWS CLI estado de la instancia. Para obtener más información, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

Descripción de las instancias de Amazon DocumentDB

Puede utilizar la Consola de administración de Amazon DocumentDB o la AWS CLI para ver detalles como puntos de conexión, grupos de seguridad VPC, entidad de certificación y grupos de parámetros relacionados con las instancias de Amazon DocumentDB.

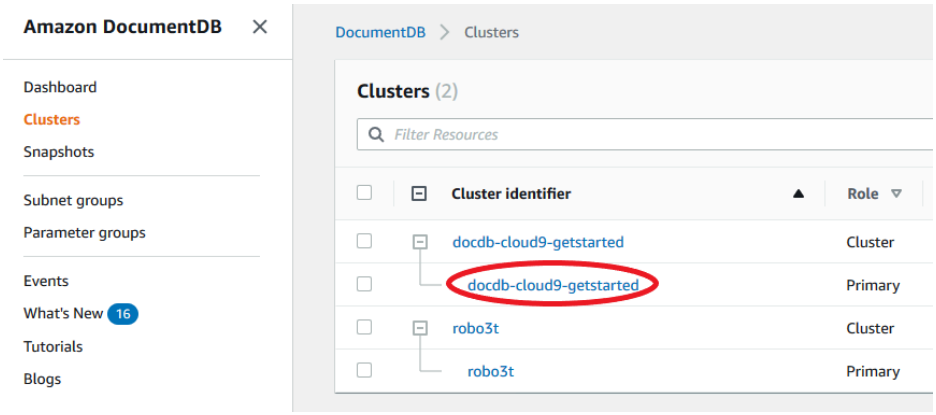
Using the Consola de administración de AWS

Para ver los detalles de las instancias mediante la Consola de administración de AWS, siga los pasos que se indican a continuación.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Clusters (Clústeres).

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En el cuadro de navegación de clústeres, verá la columna Identificador del clúster. Sus instancias se muestran en clústeres, de forma similar a la siguiente captura de pantalla.



4. En la lista de instancias, elija el nombre de la instancia cuyos detalles desea ver. La información sobre la instancia se organiza en los siguientes grupos:

   - Resumen: información general sobre la instancia, incluida la versión del motor, la clase, el estado y cualquier mantenimiento pendiente.
   - Conectividad y seguridad: la sección Conectar enumera los puntos de conexión para conectarse a esta instancia con el intérprete de comandos de mongo o con una aplicación. La sección Grupos de seguridad muestra los grupos de seguridad asociados a esta instancia y su ID de VPC y descripciones.
   - Configuración: la sección Detalles muestra las configuraciones y el estado de la instancia, incluidos el nombre de recurso de Amazon (ARN), el punto de conexión, el rol, la clase y la entidad de certificación de la instancia. También enumera la seguridad y la configuración de red de la instancia, así como la información de copia de seguridad. La sección Detalles del clúster muestra los detalles del clúster al que pertenece esta instancia. La sección Instancias del clúster muestra todas las instancias que pertenecen al clúster con el rol de cada instancia y el estado del grupo de parámetros del clúster.

     > **Note**
     >
     > Puede modificar el clúster asociado a la instancia seleccionando Modificar junto al encabezado Detalles del clúster. Para obtener más información, consulte Modificación de un clúster de Amazon DocumentDB.

   - Supervisión: CloudWatch registra las métricas de esta instancia. Para obtener más información, consulte Supervisión de Amazon DocumentDB con CloudWatch.
   - Eventos y etiquetas: la sección de eventos recientes muestra los eventos recientes de esta instancia. Amazon DocumentDB mantiene un registro de los eventos relacionados

con los clústeres, las instancias, las instantáneas, los grupos de seguridad y los grupos de parámetros de clúster. Esta información incluye la fecha, la hora y el mensaje asociados a cada evento. La sección Etiquetas muestra las etiquetas asociadas a este clúster. Para obtener más información, consulte Etiquetado de recursos de Amazon DocumentDB.

Using the AWS CLI

Para ver los detalles de sus instancias de Amazon DocumentDB mediante el AWS CLI, utilice el `describe-db-clusters` comando tal y como se muestra en los siguientes ejemplos. Para obtener más información, consulte `DescribeDBInstances` en la Referencia de la API para administración de recursos de Amazon DocumentDB.

> **Note**
>
> Para ciertas características de administración, como la administración del ciclo de vida de clúster y de instancia, Amazon DocumentDB aprovecha la tecnología operativa que se comparte con Amazon RDS. El parámetro de filtro `filterName=engine,Values=docdb` devuelve solo clústeres de Amazon DocumentDB.

1. Enumere todas las instancias de Amazon DocumentDB.

   En el siguiente AWS CLI código se enumeran los detalles de todas las instancias de Amazon DocumentDB de una región.

   Para Linux, macOS o Unix:

   ```
   aws docdb describe-db-instances \
       --filter Name=engine,Values=docdb
   ```

   Para Windows:

   ```
   aws docdb describe-db-instances \
       --filter Name=engine,Values=docdb
   ```

2. Enumeración de todos los detalles de una instancia de Amazon DocumentDB especificada

El siguiente código enumera los detalles de `sample-cluster-instance`. Incluir el parámetro `--db-instance-identifier` con el nombre de una instancia restringe la salida a la información sobre esa instancia en particular.

Para Linux, macOS o Unix:

```
aws docdb describe-db-instances \
    --db-instance-identifier sample-cluster-instance
```

Para Windows:

```
aws docdb describe-db-instances \
    --db-instance-identifier sample-cluster-instance
```

El resultado de esta operación será similar a lo que se indica a continuación:

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

Modificación de una instancia de base de datos de Amazon DocumentDB

Puede modificar su instancia de Amazon DocumentDB mediante el Consola de administración de AWS o el. AWS CLI Para modificar una instancia, la instancia debe tener el estado disponible. No puede modificar una instancia detenida. Si el clúster está detenido, inicie primero el clúster, espere a que la instancia esté disponible y, a continuación, realice las modificaciones deseadas. Para obtener más información, consulte Detención e inicio de un clúster de bases de datos de Amazon DocumentDB.

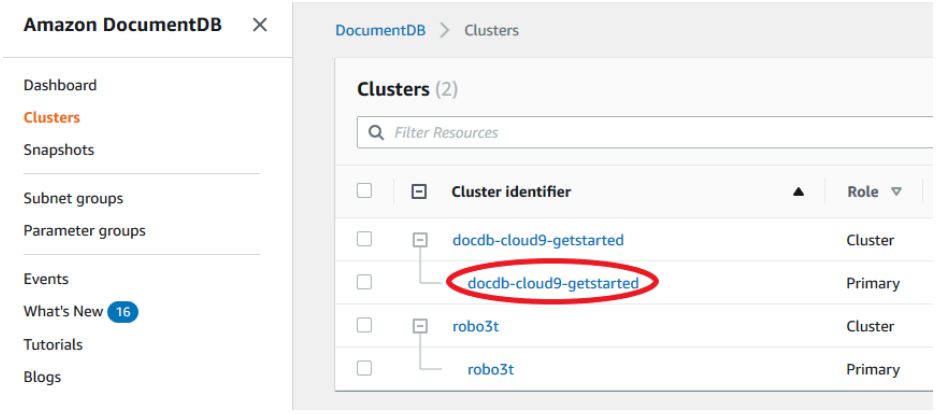
Using the Consola de administración de AWS

Complete los pasos que se indican a continuación para modificar una instancia de Amazon DocumentDB específica mediante la consola.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Clusters (Clústeres).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡)
> en la esquina superior izquierda de la página.

3. En el cuadro de navegación de clústeres, verá la columna Identificador del clúster. Sus instancias se muestran en clústeres, de forma similar a la siguiente captura de pantalla.



4. Marque la casilla de verificación situada a la izquierda de la instancia que desea modificar.
5. Elija Actions (Acciones) y después Modify (Modificar).
6. En el panel Modificar instancia: <instance-name>, realice los cambios que desee. Puede hacer los siguientes cambios:

   - Especificaciones de instancia: el identificador y la clase de la instancia. Restricciones de nombres del identificador de instancia:
     - Identificador de instancias: introduce un nombre que sea único para todas las instancias de tu propiedad Cuenta de AWS en la región actual. El identificador de instancia debe contener [1-63] caracteres alfanuméricos o guiones, tener una letra como primer carácter y no puede terminar con un guion ni contener dos guiones consecutivos.
     - Clase de instancia: en el menú desplegable, seleccione una clase de instancia para su instancia de Amazon DocumentDB. Para obtener más información, consulte Administración de clases de instancias.
   - Autoridad de certificación: certificado de servidor para esta instancia. Para obtener más información, consulte Actualización de los certificados TLS de Amazon DocumentDB.
   - Conmutación por error: durante la conmutación por error, la instancia con el nivel de promoción más alto se promoverá a primaria. Para obtener más información, consulte Conmutación por error de Amazon DocumentDB.
   - Mantenimiento : la ventana de mantenimiento en la que se aplican las modificaciones o revisiones pendientes a las instancias del clúster.
7. Cuando haya terminado, elija Continuar para ver un resumen de los cambios.
8. Después de verificar los cambios, puede aplicarlos inmediatamente o durante el siguiente período de mantenimiento en Scheduling of modifications (Programación de modificaciones).

Seleccione Modify instance (Modificar instancia) para guardar los cambios. Como alternativa, puede elegir Cancel (Cancelar) para descartar los cambios.

Los cambios pueden tardar unos minutos en aplicarse. Solo puede usar la instancia cuando su estado sea available (disponible). Puede monitorizar el estado de la instancia mediante la consola o la AWS CLI. Para obtener más información, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

Using the AWS CLI

Para modificar una instancia específica de Amazon DocumentDB mediante el AWS CLI, utilice el `modify-db-instance` con los siguientes parámetros. Para obtener más información, consulte ModifyDBInstance. El siguiente código modifica la clase de instancia a `db.r5.large` para la instancia `sample-instance`.

Parameters

- **`--db-instance-identifier`**: obligatorio. El identificador de la instancia que se va a modificar.
- **`--db-instance-class`**: opcional. La nueva capacidad de memoria e informática de la instancia (por ejemplo, `db.r5.large`). No todas las clases de instancia están disponibles en todas las Regiones de AWS. Si modifica la clase de la instancia se produce una interrupción durante el cambio. El cambio se aplica durante la siguiente ventana de mantenimiento, a menos que `ApplyImmediately` se especifique como verdadera para esta solicitud.
- **`--apply-immediately`** o **`--no-apply-immediately`**: opcional. Especifica si esta modificación debe aplicarse inmediatamente o esperar hasta el próximo periodo de mantenimiento. Si se omite este parámetro, la modificación se realiza durante el siguiente período de mantenimiento.

Example

Para Linux, macOS o Unix:

```
aws docdb modify-db-instance \
       --db-instance-identifier sample-instance \
       --db-instance-class db.r5.large \
       --apply-immediately
```

Para Windows:

```
aws docdb modify-db-instance ^
       --db-instance-identifier sample-instance ^
       --db-instance-class db.r5.large ^
       --apply-immediately
```

La salida de esta operación será similar a lo que se indica a continuación.

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

Las modificaciones pueden tardar unos minutos en aplicarse. Solo puede usar la instancia cuando su estado sea available (disponible). Puede monitorizar el estado de la instancia mediante la tecla Consola de administración de AWS o AWS CLI. Para obtener más información, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

Reinicio de la instancia de Amazon DocumentDB

Es posible que en algún momento necesite reiniciar su instancia de Amazon DocumentDB, normalmente por razones de mantenimiento. Si realiza determinados cambios, como cambiar el grupo de parámetros de clúster asociado a un clúster, debe reiniciar las instancias del clúster para

que los cambios surtan efecto. Puede reiniciar una instancia específica mediante el Consola de administración de AWS o el AWS CLI.

Cuando se reinicia una instancia, se reinicia el servicio del motor de base de datos. Al reiniciar una instancia, se produce una interrupción momentánea, durante la cual su estado se establece en `rebooting`. Cuando finaliza el reinicio, se crea un evento de Amazon DocumentDB.

El reinicio de una instancia no produce una conmutación por error. Para realizar la conmutación por error de un clúster de Amazon DocumentDB, utilice Consola de administración de AWS la operación o AWS CLI . `failover-db-cluster` Para obtener más información, consulte Conmutación por error de Amazon DocumentDB.

No puede reiniciar su instancia si no tiene el estado disponible. La base de datos puede no estar disponible por varias razones, como una modificación solicitada anteriormente o una acción durante un periodo de mantenimiento. Para obtener más información acerca de los estados de las instancias, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

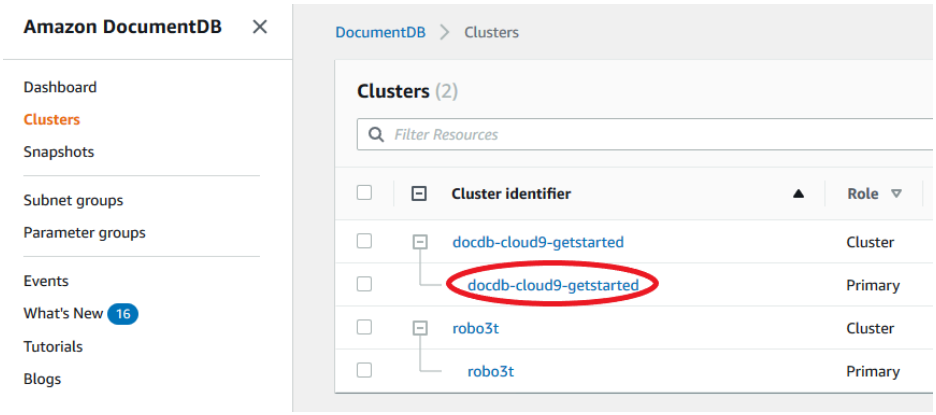
Using the Consola de administración de AWS

El siguiente procedimiento reinicia una instancia especificada mediante la consola.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Clusters (Clústeres).

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En el cuadro de navegación de clústeres, verá la columna Identificador del clúster. Sus instancias se muestran en clústeres, de forma similar a la siguiente captura de pantalla.



4. Marque la casilla de verificación situada a la izquierda de la instancia que desea reiniciar.
5. Elija Actions (Acciones), después Reboot (Reiniciar) y, por último, Reboot (Reiniciar) para confirmar el reinicio.

El reinicio de la instancia puede tardar unos minutos. Solo puede usar la instancia cuando su estado sea available (disponible). Puede monitorizar el estado de la instancia mediante la consola o la AWS CLI. Para obtener más información, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

Using the AWS CLI

Para reiniciar una instancia de Amazon DocumentDB, utilice la operación `reboot-db-instance` con el parámetro `--db-instance-identifier`. Este parámetro especifica el identificador de la instancia que se va a reiniciar.

El siguiente código reinicia la instancia `sample-instance`.

Example

Para Linux, macOS o Unix:

```
aws docdb reboot-db-instance \
       --db-instance-identifier sample-instance
```

Para Windows:

```
aws docdb reboot-db-instance ^
       --db-instance-identifier sample-instance
```

La salida de esta operación será similar a lo que se indica a continuación.

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

El reinicio de la instancia puede tardar unos minutos. Solo puede usar la instancia cuando su estado sea available (disponible). Puede monitorizar el estado de la instancia mediante la consola o la AWS CLI. Para obtener más información, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

Eliminación de una instancia de Amazon DocumentDB

Puede eliminar su instancia de Amazon DocumentDB mediante el Consola de administración de AWS o el. AWS CLI Para eliminar una instancia, la instancia debe estar en el estado disponible. No puede eliminar una instancia detenida. Si el clúster de Amazon DocumentDB que contiene la instancia está detenido, inicie primero el clúster, espere a que la instancia esté disponible y, a continuación, elimine la instancia. Para obtener más información, consulte Detención e inicio de un clúster de bases de datos de Amazon DocumentDB.

> **Note**
>
> Amazon DocumentDB almacena todos los datos en el volumen del clúster. Los datos se conservan en ese volumen de clúster, incluso si elimina todas las instancias del clúster. Si necesita obtener acceso a los datos de nuevo, puede agregar una instancia al clúster en cualquier momento y reanudar el trabajo donde lo dejó.

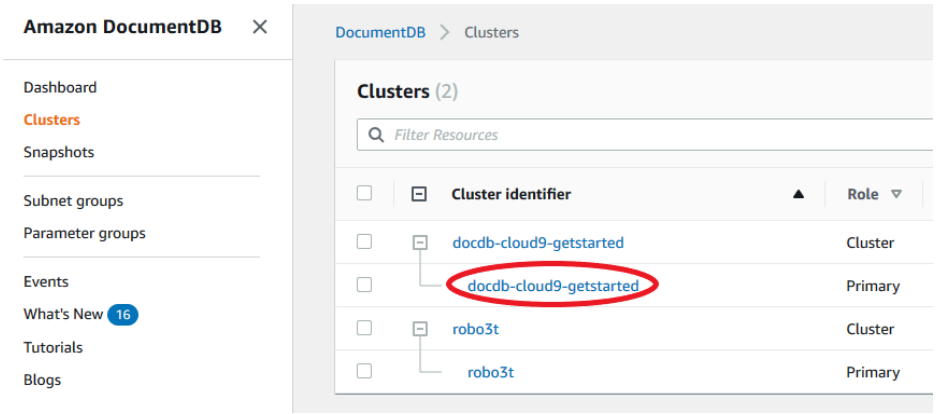
Using the Consola de administración de AWS

El siguiente procedimiento elimina una instancia de Amazon DocumentDB especificada mediante la consola.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Clusters (Clústeres).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En el cuadro de navegación de clústeres, verá la columna Identificador del clúster. Sus instancias se muestran en clústeres, de forma similar a la siguiente captura de pantalla.



4. Marque la casilla de verificación situada a la izquierda de la instancia que desea eliminar.
5. Seleccione Acciones y, a continuación, elija Eliminar.

   1. Si va a eliminar la última instancia del clúster:
      - Create final cluster snapshot? (¿Crear instantánea de clúster final?) Elija Sí si desea crear una instantánea final antes de eliminar el clúster. En caso contrario, elija No.
      - Nombre de la instantánea final: si ha decidido crear una instantánea final, escriba el identificador de la instantánea de clúster o de la nueva instantánea de clúster que se ha creado.
      - Delete <instance-name> instance? (¿Eliminar <nombre-de-la-instancia>?) Escriba la frase Eliminar clúster completo en el campo para confirmar la eliminación.

2. Si no va a eliminar la última instancia del clúster:
   - Delete <instance-name> instance? (¿Eliminar <nombre-de-la-instancia>?) Escriba la frase Eliminarme en el campo para confirmar la eliminación.

6. Seleccione Delete (Eliminar) para eliminar la instancia.

La eliminación de una instancia tarda varios minutos. Para monitorear el estado de una instancia, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

Using the AWS CLI

El siguiente procedimiento elimina una instancia de Amazon DocumentDB mediante la AWS CLI.

1. En primer lugar, determine cuántas instancias hay en el clúster de Amazon DocumentDB:

   Para determinar cuántas instancias hay en el clúster, ejecute el comando `describe-db-clusters` de la siguiente manera.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].
[DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
```

   La salida de esta operación será similar a lo que se indica a continuación.

```
[
    [
        "sample-cluster",
        [
            "sample-instance-1",
            "sample-instance-2"
        ]
    ]
]
```

2. Si hay más de una instancia en el clúster de Amazon DocumentDB:

   Para eliminar una instancia de Amazon DocumentDB especificada, utilice el comando `delete-db-instance` con el parámetro `--db-instance-identifier`, como se muestra a continuación. La eliminación de una instancia tarda varios minutos. Para monitorear el estado de una instancia, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

```
aws docdb delete-db-instance \
       --db-instance-identifier sample-instance-2
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "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. Si la instancia que desea eliminar es la última instancia del clúster de Amazon DocumentDB:

   Si elimina la última instancia de un clúster de Amazon DocumentDB, también elimina ese clúster, así como las instantáneas automáticas y las copias de seguridad continuas asociadas a ese clúster.

   Para eliminar la última instancia del clúster, puede eliminar el clúster y, si lo desea, crear una instantánea final. Para obtener más información, consulte Eliminación de un clúster de Amazon DocumentDB.

Protección contra eliminación

Al eliminar la última instancia de un clúster de Amazon DocumentDB, también se eliminará el clúster, así como las instantáneas automáticas y las copias de seguridad continuas asociadas a ese clúster. Amazon DocumentDB aplica la protección contra la eliminación de un clúster, independientemente de si realiza la operación de eliminación con o con. Consola de administración de AWS AWS CLI Si la protección contra eliminación está habilitada, no podrá eliminar un clúster.

Para eliminar un clúster con la protección contra eliminación habilitada, primero debe modificar el clúster y deshabilitar la protección contra eliminación. Para obtener más información, consulte Eliminación de un clúster de Amazon DocumentDB.

Administración de clases de instancias

La clase de instancia de base de datos determina la capacidad de cómputo y de memoria de una instancia de Amazon DocumentDB (compatible con MongoDB). La clase de instancia que necesita depende de la potencia de procesamiento y de los requisitos de memoria.

Amazon DocumentDB admite las familias de clases de instancias R4, R5, R6G, R8G, T3 y T4G. Se trata de clases de instancias de la generación actual que se han optimizado para las aplicaciones que hacen un uso intensivo de la memoria. Para conocer las especificaciones de estas clases, consulte Especificaciones de clases de instancias.

Temas

- Determinación de la clase de una instancia
- Cambio de la clase de una instancia
- Clases de instancias admitidas por región
- Especificaciones de clases de instancias

Determinación de la clase de una instancia

Para determinar la clase de una instancia, puedes usar la `describe-db-instances` AWS CLI operación Consola de administración de AWS o.

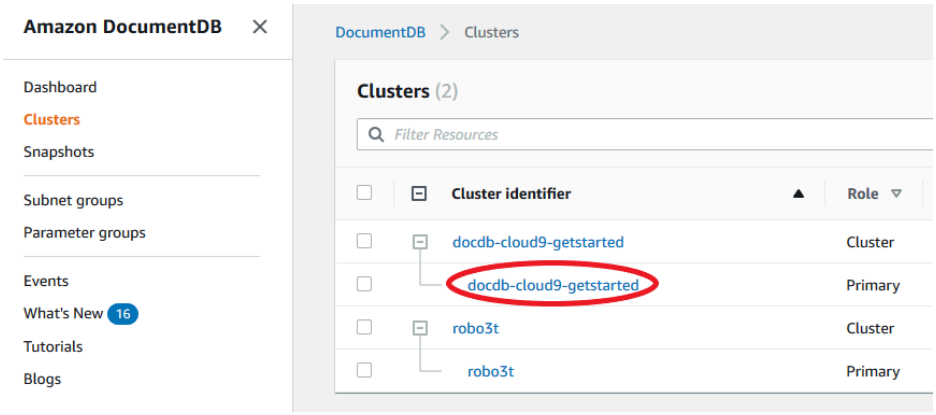
Using the Consola de administración de AWS

Para determinar la clase de instancia de las instancias del clúster, complete los siguientes pasos en la consola.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. Para encontrar la instancia que le interesa, elija Clústeres en el panel de navegación.

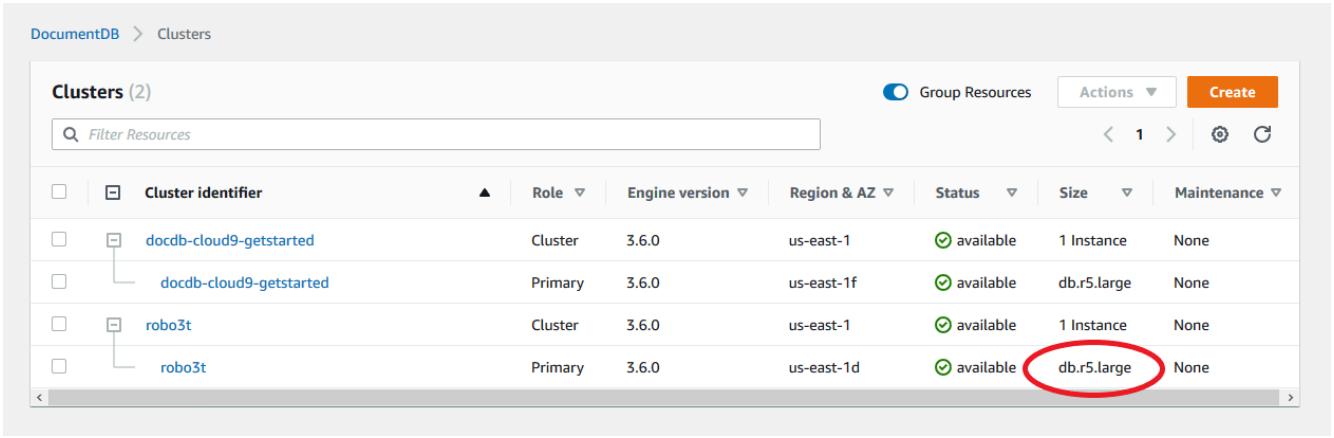
> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En el cuadro de navegación de clústeres, verá la columna Identificador del clúster. Sus instancias se enumeran en clústeres, de forma similar a la siguiente captura de pantalla.



4. En la lista de instancias, expande el clúster para encontrar las instancias que le interesan. Encuentre la instancia que desee. A continuación, fíjese en la columna Tamaño de la fila correspondiente a la instancia.

   En la imagen siguiente, la clase de la instancia `robo3t` es `db.r5.4xlarge`.



Using the AWS CLI

Para determinar la clase de una instancia mediante el AWS CLI, usa la `describe-db-instances` operación con los siguientes parámetros.

- **`--db-instance-identifier`**— Opcional. Especifica la instancia para la que desea encontrar la clase de instancia. Si se omite este parámetro, `describe-db-instances` devuelve una descripción para un máximo de 100 instancias.
- **`--query`**: opcional. Especifica los miembros de la instancia que se van a incluir en los resultados. Si se omite este parámetro, se devuelven todos los miembros de la instancia.

Example

En el siguiente ejemplo se busca el nombre y la clase de la instancia para la instancia `sample-instance-1`.

Para Linux, macOS o Unix:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' \
    --db-instance-identifier sample-instance-1
```

Para Windows:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' ^
    --db-instance-identifier sample-instance-1
```

La salida de esta operación será similar a lo que se indica a continuación.

```
[
    [
        "sample-instance-1",
        "db.r5.large"
    ]
```

Example

En el siguiente ejemplo se busca el nombre y la clase de la instancia para un máximo de 100 instancias de Amazon DocumentDB.

Para Linux, macOS o Unix:

```
aws docdb describe-db-instances \
```

```
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' \
    --filter Name=engine,Values=docdb
```

Para Windows:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' ^
    --filter Name=engine,Values=docdb
```

La salida de esta operación será similar a lo que se indica a continuación.

```
[
    [
        "sample-instance-1",
        "db.r5.large"
    ],
    [
        "sample-instance-2",
        "db.r5.large"
    ],
    [
        "sample-instance-3",
        "db.r5.4xlarge"
    ],
    [
        "sample-instance-4",
        "db.r5.4xlarge"
    ]
]
```

Para obtener más información, consulte Descripción de las instancias de Amazon DocumentDB.

Cambio de la clase de una instancia

Puedes cambiar la clase de instancia de tu instancia usando la Consola de administración de AWS o la AWS CLI. Para obtener más información, consulte Modificación de una instancia de base de datos de Amazon DocumentDB.

Clases de instancias admitidas por región

Amazon DocumentDB admite las siguientes clases de instancia:

- R8G—La última generación de instancias optimizadas para la memoria equipadas con procesadores Arm-based AWS Graviton4 que proporcionan un rendimiento hasta un 30% superior al de las instancias R6G.
- R6G— Memory-optimized instancias equipadas con procesadores Arm-based AWS Graviton2 que proporcionan hasta un 30% más de rendimiento que las instancias R5 a un costo un 5% menor.
- R6GD— Instancias Memory-optimized R6G con almacenamiento en Solid-State disco (SSD) local basado en memoria no volátil express (NVMe) para datos efímeros.
- R5— Memory-optimized instancias que ofrecen un rendimiento hasta un 100% mejor que las instancias R4 por el mismo coste de instancia.
- R4: instancias optimizadas para memoria de generación anterior.
- T4G— un tipo de instancia de uso general, de Latest-generation bajo costo, con procesadores Arm-based AWS Graviton2, que proporciona un nivel básico de rendimiento de la CPU, ofrece una relación precio-rendimiento hasta un 35% más alta que las instancias T3 y es ideal para ejecutar aplicaciones con un uso moderado de la CPU que experimenten picos temporales de uso.
- T3: tipo de instancia de uso general y ampliables que proporcionan un nivel básico de rendimiento de la CPU con posibilidad de ampliar el uso de la CPU en cualquier momento durante el tiempo que sea necesario.

Para conocer las especificaciones detalladas de las clases de instancias, consulte Especificaciones de clases de instancias.

> **Note**
>
> I/O-Optimized el almacenamiento solo está disponible en las versiones 5.0 y 8.0 del motor (clústeres basados en instancias).

Una clase de instancia determinada se podría admitir o no en una región determinada. En la siguiente tabla, se especifica qué clases de instancias admite Amazon DocumentDB en cada región.

Clases de instancias admitidas por región

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Sin servidor
Este de EE. UU. (Ohio)	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	compatible
Este de EE. UU. (Norte de Virginia)	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	compatible
Oeste de EE. UU. (Oregón)	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	compatible
África (Ciudad del Cabo)			Soportado	Soportado		Soportado	Soportado	compatible
América del Sur (São Paulo)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Hong Kong)	Soportado		Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Hyderabad)			Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Malasia)			Soportado			Soportado	Soportado	compatible
Asia-Pacífico (Mumbai)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Osaka)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Seúl)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Sin servidor
Asia-Pacífico (Sídney)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Yakarta)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Melbourne)			Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Singapur)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Asia-Pacífico (Tailandia)			Soportado			Soportado	Soportado	compatible
Asia-Pacífico (Tokio)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Canadá (centro)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Oeste de Canadá (Calgary)			Soportado	Soportado		Soportado	Soportado	compatible
Europa (Fráncfort)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Europa (Zúrich)		Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Europa (Irlanda)	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	compatible
Europa (Londres)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Sin servidor
Europa (Milán)			Soportado	Soportado		Soportado	Soportado	compatible
Europa (París)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Europa (España)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
Europa (Estocolmo)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
México (centro)			Soportado			Soportado	Soportado	compatible
Medio Oriente (EAU)			Soportado	Soportado		Soportado	Soportado	compatible
China (Pekín)	Soportado	Soportado	Soportado	Soportado		Soportado	Soportado	compatible
China (Ningxia)	Soportado		Soportado	Soportado		Soportado	Soportado	compatible
Israel (Tel Aviv)			Soportado	Soportado		Soportado	Soportado	Soportado
AWS GovCloud (US-West)	Soportado	Soportado	Soportado	Soportado			Soportado	Soportado
AWS GovCloud (US-East)		Soportado	Soportado	Soportado		Soportado	Soportado	compatible

Especificaciones de clases de instancias

La siguiente tabla proporciona información detallada de las clases de instancias de Amazon DocumentDB, incluidos los tipos de instancias que se admiten en cada clase. Puede encontrar explicaciones sobre cada columna de la tabla debajo de la tabla.

Clase de instancia	CPU virtuales[1]	Memoria (GiB)[2]	Caché por niveles SSD NVMe (GiB)[3]	Temperatura máxima de almacenamiento (GiB) 4	Ancho de banda de base/con ampliación (Gbps)[5]	Motores compatibles[6]
R8G: clase de Memory-Optimized instancia de generación actual basada en Graviton4 El R8G es la familia de instancias más reciente (Graviton4) y solo está disponible en las versiones 5.0 y 8.0 del motor.						
`db.r8g.large`	2	16	-	30	0,937/12,5	5.0.0 y 8.0.0
`db.r8g.xlarge`	4	32	-	60	1,875/12,5	5.0.0 y 8.0.0
`db.r8g.2xlarge`	8	64	-	121	3,75/15,0	5.0.0 y 8.0.0
`db.r8g.4xlarge`	16	128	-	243	7,5/15,0	5.0.0 y 8.0.0
`db.r8g.8xlarge`	32	256	-	488	15	5.0.0 y 8.0.0
`db.r8g.12xlarge`	48	384	-	732	22	5.0.0 y 8.0.0
`db.r8g.16xlarge`	64	512	-	987	30	5.0.0 y 8.0.0

Clase de instancia	CPU virtuales[1]	Memoria (GiB)[2]	Caché por niveles SSD NVMe (GiB)[3]	Temperatura máxima de almacenamiento (GiB) 4	Ancho de banda de base/con ampliación (Gbps)[5]	Motores compatibles[6]
db.r8g.24xlarge	96	768	-	1484	40	5.0.0 y 8.0.0
db.r8g.48xlarge	192	1536	-	2967	50	5.0.0 y 8.0.0
R6G: clase de Memory-Optimized instancia de generación actual basada en Graviton2						
db.r6g.large	2	16	-	32	0.75/10	4.0.0, 5.0.0 y 8.0.0
db.r6g.xlarge	4	32	-	63	1,25/10	4.0.0, 5.0.0 y 8.0.0
db.r6g.2xlarge	8	64	-	126	2.5/10	4.0.0, 5.0.0 y 8.0.0
db.r6g.4xlarge	16	128	-	252	5.0/10	4.0.0, 5.0.0 y 8.0.0
db.r6g.8xlarge	32	256	-	504	12	4.0.0, 5.0.0 y 8.0.0
db.r6g.12xlarge	48	384	-	756	20	4.0.0, 5.0.0 y 8.0.0
db.r6g.16xlarge	64	512	-	1008	25	4.0.0, 5.0.0 y 8.0.0
R6GD: clase de instancia de generación actual basada en Graviton2 NVMe-backed						

Clase de instancia	CPU virtuales[1]	Memoria (GiB)[2]	Caché por niveles SSD NVMe (GiB)[3]	Temperatura máxima de almacenamiento (GiB)[4]	Ancho de banda de base/con ampliación (Gbps)[5]	Motores compatibles[6]
db.r6gd.xlarge	4	32	173	64	1.25/10	5.0.0 y 8.0.0
db.r6gd.2xlarge	8	64	346	128	2.5/10	5.0.0 y 8.0.0
db.r6gd.4xlarge	16	128	694	256	5.0/10	5.0.0 y 8.0.0
db.r6gd.8xlarge	32	256	1388	512	12	5.0.0 y 8.0.0
db.r6gd.12xlarge	48	384	2082	768	20	5.0.0 y 8.0.0
db.r6gd.16xlarge	64	512	2776	1024	25	5.0.0 y 8.0.0
R5: clase de instancia de la generación anterior Memory-Optimized						
db.r5.large	2	16	-	31	0.75/10	3.6.0, 4.0.0, 5.0.0 y 8.0.0
db.r5.xlarge	4	32	-	62	1,25/10	3.6.0, 4.0.0, 5.0.0 y 8.0.0
db.r5.2xlarge	8	64	-	124	2.5/10	3.6.0, 4.0.0, 5.0.0 y 8.0.0

Clase de instancia	CPU virtuales[1]	Memoria (GiB)[2]	Caché por niveles SSD NVMe (GiB)[3]	Temperatura máxima de almacenamiento (GiB)[4]	Ancho de banda de base/con ampliación (Gbps)[5]	Motores compatibles[6]
db.r5.4xlarge	16	128	-	249	5.0/10	3.6.0, 4.0.0, 5.0.0 y 8.0.0
db.r5.8xlarge	32	256	-	504	10	3.6.0, 4.0.0, 5.0.0 y 8.0.0
db.r5.12xlarge	48	384	-	748	12	3.6.0, 4.0.0, 5.0.0 y 8.0.0
db.r5.16xlarge	64	512	-	1008	20	3.6.0, 4.0.0, 5.0.0 y 8.0.0
db.r5.24xlarge	96	768	-	1500	25	3.6.0, 4.0.0, 5.0.0 y 8.0.0
R4: clase de instancia de la generación anterior Memory-Optimized Las instancias R4 solo son compatibles con la versión 3.6 del motor. Tenga en cuenta que Amazon DocumentDB 3.6 finaliza el soporte estándar el 30 de marzo de 2026. El soporte extendido (de pago) está disponible hasta marzo de 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

Clase de instancia	CPU virtuales[1]	Memoria (GiB)[2]	Caché por niveles SSD NVMe (GiB)[3]	Temperatura máxima de almacenamiento (GiB)[4]	Ancho de banda de base/con ampliación (Gbps)[5]	Motores compatibles[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: clases de instancia de rendimiento ampliable de última generación basadas en Graviton2 T-series las instancias se ejecutan en modo de ráfaga de CPU ilimitada. El uso de ráfagas más allá de la línea de base se factura de forma adicional. No es compatible con los clústeres globales.						
db.t4g.medium	2	4	-	8.13	0,256/5	4.0.0, 5.0.0 y 8.0.0
T3: clases de instancia de rendimiento ampliable de generación anterior						
db.t3.medium	2	4	-	7.5	0,256/5	3.6.0, 4.0.0, 5.0.0 y 8.0.0

Clase de instancia	CPU virtuales[1]	Memoria (GiB)[2]	Caché por niveles SSD NVMe (GiB)[3]	Temperatura máxima de almacenamiento (GiB)[4]	Ancho de banda de base/con ampliación (Gbps)[5]	Motores compatibles[6]

1. vCPU: el número de unidades de procesamiento central (CPU) virtuales. Una CPU virtual es una unidad de capacidad que se puede usar para comparar clases de instancias. En lugar de comprar o arrendar un procesamiento concreto para usarlo durante varios meses o años, la capacidad se alquila por horas. Nuestro objetivo es proporcionar una cantidad constante de capacidad de CPU sea cual sea el hardware subyacente.
2. Memoria (GiB): RAM, en gigabytes, que se asigna a la instancia. A menudo, hay una relación coherente entre memoria y vCPU.
3. Caché por niveles SSD NVMe: el espacio del volumen de la SSD, medido en gigabytes, asignado como caché extendida para almacenar datos efímeros. Esta caché solo está disponible en instancias. NVMe-backed
4. Temperatura máxima de almacenamiento (GiB): el espacio, medido en gigabytes, que se asigna a la instancia para el almacenamiento temporal no persistente de archivos. Por NVMe-backed ejemplo, este almacenamiento se aloja en un volumen NVMe-based SSD. En todas las demás instancias, se aloja en Amazon Elastic Block Store (EBS).
5. Ancho de banda de base/con ampliación (Gbps): el ancho de banda con ampliación representa el ancho de banda máximo en gigabits por segundo. Divídalo entre 8 para obtener el rendimiento esperado en gigabytes por segundo. Las instancias de tamaño 4xlarge o menos tienen un ancho de banda de base. Para satisfacer una demanda adicional, pueden utilizar un mecanismo de I/O crédito de red para superar su ancho de banda básico. Las instancias pueden utilizar la banda ancha de fragmentación durante un tiempo limitado, normalmente de 5 a 60 minutos, en función del tamaño de la instancia.
6. Motores compatibles: los motores de Amazon DocumentDB que admiten la clase de instancia.

NVMe-backed instancias

NVMe-backed las instancias ofrecen un rendimiento de consultas hasta 7 veces más rápido para aplicaciones con grandes conjuntos de datos que superan la memoria de una instancia normal. Estas instancias aprovechan el almacenamiento en unidades (SSD) local no volátil basado en Solid-State

Memory Express (NVMe) disponible en las instancias r6gd para almacenar datos efímeros, reducir el acceso al almacenamiento basado en la red y mejorar la latencia de lectura y el rendimiento.

El espacio de la SSD local se divide en dos secciones:

- Caché por niveles: aproximadamente el 73 % de la SSD local se asigna como caché de base de datos, lo que permite al sistema almacenar hasta cinco veces más páginas de base de datos que la memoria principal por sí sola. La SSD local actúa como caché de segundo nivel, mientras que la caché de búfer en memoria existente sigue siendo la caché de primer nivel. La consulta accede solamente al almacenamiento externo si se produce un error tanto en la caché de búfer como en la de la SSD.
- Almacenamiento temporal: el 27 % restante se reserva para el almacenamiento temporal no persistente de archivos, que se utiliza para consultas complejas que implican ordenaciones u operaciones con consumo intensivo de recursos, como la compilación de índices. En instancias normales, el espacio temporal se encuentra en un volumen de Amazon Elastic Block Store (EBS). El almacenamiento temporal alojado localmente en la SSD reduce hasta dos veces la latencia de las consultas que implican ordenaciones y acelera las operaciones con consumo intensivo de recursos, como la compilación de índices.

Encontrará más información sobre el tipo de NVMe-backed instancias admitidas y sus especificaciones en. Especificaciones de clases de instancias

Temas

- Casos de uso recomendados para NVMe-backed las instancias
- Uso de instancias con Amazon DocumentDB NVMe-backed
- Monitorización de instancias NVMe-backed

Casos de uso recomendados para NVMe-backed las instancias

Usa NVMe-backed instancias en los siguientes escenarios:

- Read-heavy cargas de trabajo: si tu carga de trabajo requiere mucha lectura y tu conjunto de datos supera el tamaño de la caché del búfer (lo que indican `ReadIOPS` las métricas altas `BufferCacheHitRatio` y bajas), las NVMe-backed instancias pueden ofrecer beneficios de rendimiento.

- Update-heavy cargas de trabajo: si tu carga de trabajo requiere muchas actualizaciones y la recolección de basura no puede mantenerse al día debido a la latencia de lectura del almacenamiento en red, las NVMe-backed instancias podrían ayudar a mitigar el problema.

NVMe-backed las instancias pueden beneficiar a varios casos de uso, entre los que se incluyen:

- Internet-scale aplicaciones: las aplicaciones como el procesamiento de pagos, la facturación y el comercio electrónico con acuerdos de nivel de servicio (SLA) de rendimiento estrictos pueden aprovechar las ventajas de rendimiento de las NVMe-backed instancias.
- Real-time paneles de generación de informes: los paneles que ejecutan cientos de consultas para su metrics/data recopilación pueden beneficiarse de la baja latencia y el alto rendimiento de las instancias. NVMe-backed
- Aplicaciones de IA generativa: las aplicaciones que utilizan la búsqueda vectorial para encontrar los vecinos exactos o más cercanos en millones de incrustaciones vectoriales pueden aprovechar el alto rendimiento de las instancias. NVMe-backed

Uso de instancias con Amazon DocumentDB NVMe-backed

Para usar NVMe-backed instancias de Amazon DocumentDB:

- Cree un clúster de Amazon DocumentDB y añada una de las clases de NVMe-backed instancias. Para obtener más información, consulte Creación de un clúster de Amazon DocumentDB.
- Como alternativa, modifique un clúster de Amazon DocumentDB existente para usar una de las clases de NVMe-backed instancias. Para obtener más información, consulte Modificación de un clúster de Amazon DocumentDB.

Para comprobar la disponibilidad de las NVMe-backed instancias en diferentes AWS regiones, consulteClases de instancias admitidas por región.

Si desea volver de una NVMe-backed instancia a una instancia normal, modifique la clase de instancia de base de datos de su instancia de Amazon DocumentDB por una clase de instancia similar sin el almacenamiento NVMe. Por ejemplo, si la clase de instancia actual es «db.r6gd.4xlarge», elija «db.r6g.4xlarge» para volver atrás. Para obtener más información, consulte Modificación de un clúster de Amazon DocumentDB.

Monitorización de instancias NVMe-backed

Además de las métricas de instancia habituales disponibles en Amazon CloudWatch, las NVMe-backed instancias emiten métricas adicionales específicas para el almacenamiento NVMe-based SSD, las IOPS y el rendimiento.

```
NVMeStorageCacheHitRatio
FreeNVMeStorage
ReadIOPSNVMeStorage
ReadLatencyNVMeStorage
ReadThroughputNVMeStorage
WriteIOPSNVMeStorage
WriteLatencyNVMeStorage
WriteThroughputNVMeStorage
```

Para obtener más información sobre estas métricas, consulte NVMe-backed métricas de instancia

Administración de grupos de subredes de Amazon DocumentDB

Una nube privada virtual (VPC) es una red virtual dedicada a su Cuenta de AWS. Está aislada lógicamente de otras redes virtuales en la AWS nube. Puede lanzar sus AWS recursos, como los clústeres de Amazon DocumentDB, en su Amazon VPC. Puede especificar un intervalo de direcciones IP para la VPC, añadir subredes, asociar grupos de seguridad y configurar tablas de ruteo.

Una subred es un rango de direcciones IP en su VPC de Amazon. Puede lanzar AWS recursos en una subred específica. Utilice una subred pública para los recursos que deben conectarse a Internet. Utilice una subred privada para los recursos que no dispondrán de conexión a Internet. Para obtener más información sobre subredes públicas y privadas, consulte Conceptos básicos de VPC y subredes en la Guía de usuario de la Nube privada virtual de Amazon.

Un grupo de subredes de base de datos es una colección de subredes que se crean en una VPC y que después se asignan a los clústeres. Un grupo de subredes le permite especificar una VPC determinada a la hora de crear clústeres. Si utiliza el grupo de subredes `default`, este abarca todas las subredes de la VPC.

Cada grupo de subredes de base de datos debe tener subredes como mínimo en dos zonas de disponibilidad de una región de determinada. Cuando crea un clúster de base de datos en una VPC, debe elegir un grupo de subredes de base de datos. Amazon DocumentDB utiliza ese grupo de

subredes de base de datos y su zona de disponibilidad preferida para seleccionar una subred y una dirección IP dentro de esa subred con el fin de asociarla al clúster. Si se produce un error en la instancia principal, Amazon DocumentDB puede promover una instancia de réplica correspondiente para que se convierta en la nueva instancia principal. A continuación, puede crear una nueva instancia de réplica utilizando una dirección IP de la subred en la que se encontraba la instancia principal anterior.

Cuando Amazon DocumentDB crea una instancia en una VPC, asigna una interfaz de red al clúster utilizando la dirección IP seleccionada del grupo de subredes de base de datos. Utilice siempre el nombre DNS porque la dirección IP subyacente puede cambiar durante la conmutación por error. Para obtener más información, consulte Puntos de conexión de Amazon DocumentDB.

Para obtener información sobre la creación de su propia VPC y sus subredes, consulte Uso de VPC y subredes en la Guía del usuario de la nube privada virtual de Amazon.

Temas

- Creación de un grupo de subredes de Amazon DocumentDB
- Descripción de un grupo de subredes de Amazon DocumentDB
- Modificación de un grupo de subredes de Amazon DocumentDB
- Eliminación de un grupo de subredes de Amazon DocumentDB

Creación de un grupo de subredes de Amazon DocumentDB

Al crear un clúster de Amazon DocumentDB, debe elegir una VPC de Amazon y el grupo de subredes correspondiente dentro de la VPC de Amazon para lanzar el clúster. Las subredes determinan la zona de disponibilidad y el rango de direcciones IP de la zona de disponibilidad que se deben utilizar para lanzar una instancia.

Un subgrupo de subredes es un determinado grupo de subredes (o AZ) que le permite especificar las zonas de disponibilidad que desea utilizar para lanzar instancias de Amazon DocumentDB. Por ejemplo, en un clúster con tres instancias, se recomienda que cada una de dichas instancias esté aprovisionada en una zona de disponibilidad (AZ) distinta, al hacerlo, optimiza la alta disponibilidad. De esa forma, si una de las AZ falla, solo se verá afectada una sola instancia.

Las instancias de Amazon DocumentDB se pueden aprovisionar en hasta tres zonas de disponibilidad. Aunque un grupo de subredes tenga más de tres subredes, solo podrá utilizar tres de esas subredes para crear un clúster de Amazon DocumentDB. Al crear un grupo de subredes, elija solo las tres subredes en las que desea implementar las instancias.

Por ejemplo: se crea un clúster y Amazon DocumentDB elige las AZ {1A, 1B y 1C}. Si intenta crear una instancia en la AZ {1D}, la llamada a la API no funcionará correctamente. Sin embargo, si decide crear una instancia sin especificar la zona de disponibilidad concreta, Amazon DocumentDB elegirá una zona de disponibilidad por usted. Amazon DocumentDB utiliza un algoritmo para equilibrar la carga de las instancias en las AZ para ayudarle a obtener una alta disponibilidad. Si se aprovisionan tres instancias, de forma predeterminada se repartirán entre tres zonas de disponibilidad en lugar de aprovisionar todas ellas en una única AZ.

Prácticas recomendadas

- A menos que tenga un motivo específico, cree siempre un grupo de subredes con tres subredes. Lo anterior asegurará que los clústeres con tres o más instancias podrán lograr una mayor disponibilidad, debido a que las instancias se aprovisionarán en tres zonas de disponibilidad.
- Reparta siempre las instancias en varias zonas de disponibilidad para lograr una alta disponibilidad. Nunca ponga todas las instancias de un clúster en una única zona de disponibilidad.
- Debido a que pueden producirse eventos de conmutación por error en cualquier momento, no debe dar por hecho que una instancia principal o las instancias de réplica siempre estarán en una AZ determinada.

Cómo crear un grupo de subredes

Puede usar el Consola de administración de AWS o AWS CLI para crear un grupo de subredes de Amazon DocumentDB:

Using the Consola de administración de AWS

Siga los siguientes pasos para crear un grupo de subredes de Amazon DocumentDB.

Creación de un grupo de subredes de Amazon DocumentDB

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, elija Subnet groups (Grupos de subredes)y, a continuación, elija Create (Crear).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En la página Create subnet group (Crear grupo de subredes):

   a. En la sección Subnet group details (Detalles del grupo de subredes):

      i. Nombre: introduzca un nombre significativo para el grupo de la subred.

      ii. Description (Descripción): introduzca una descripción del grupo de subredes.

   b. En la sección Add subnets (Añadir subredes):

      i. VPC: elija una VPC de la lista, para este grupo de subredes.

      ii. Realice una de las siguientes acciones:

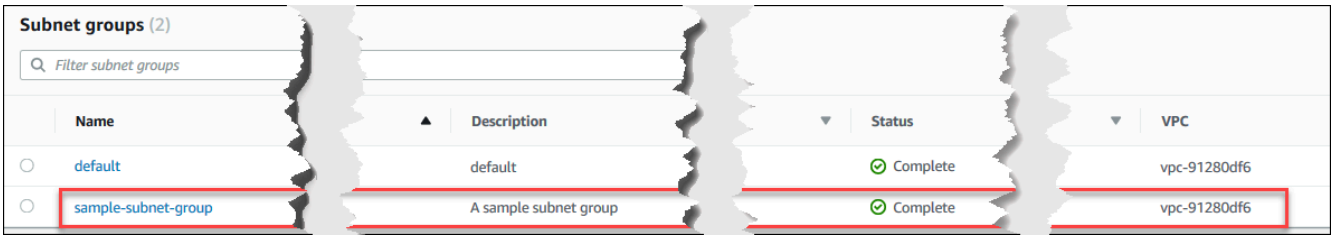
      - Para incluir todas las subredes en la VPC seleccionada, elija Add all the subnets related to this VPC (Añadir todas las subredes asociadas a esta VPC).
      - Para especificar subredes para este grupo de subredes, realice los pasos siguientes para cada zona de disponibilidad para la que desee incluir subredes. Debe incluir al menos dos zonas de disponibilidad.

        A. Zona de disponibilidad: de la lista, elija una zona de disponibilidad.

        B. Subred: de la lista, elija una subred de la zona de disponibilidad elegida para este grupo de subredes.

        C. Seleccione Add subnet (Añadir subred).

4. Seleccione Crear. Cuando se crea el grupo de subredes, se muestra junto con los demás grupos de subredes.



Using the AWS CLI

Antes de poder crear un grupo de subredes con el AWS CLI, primero debe determinar qué subredes están disponibles. Ejecute la siguiente AWS CLI operación para ver una lista de las zonas de disponibilidad y sus subredes.

Parámetros:

- **`--db-subnet-group`**: opcional. Si especifica un grupo de subredes concreto, puede ver las zonas de disponibilidad y las subredes de dicho grupo. Si omite este parámetro, puede ver las zonas de disponibilidad y las subredes de todos los grupos de subredes. Si especifica el grupo de subredes `default` puede ver todas las subredes de la VPC.

Example

Para Linux, macOS o Unix:

```
aws docdb describe-db-subnet-groups \
    --db-subnet-group-name default \
    --query 'DBSubnetGroups[*].[DBSubnetGroupName,Subnets[*].
[SubnetAvailabilityZone.Name,SubnetIdentifier]]'
```

Para Windows:

```
aws docdb describe-db-subnet-groups ^
    --db-subnet-group-name default ^
    --query 'DBSubnetGroups[*].[DBSubnetGroupName,Subnets[*].
[SubnetAvailabilityZone.Name,SubnetIdentifier]]'
```

La salida de esta operación será similar a lo que se indica a continuación (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"
            ]
        ]
    ]
]
```

Si utiliza el resultado de la operación anterior, puede crear un nuevo grupo de subredes. El grupo de subredes nuevo debe incluir subredes de al menos dos zonas de disponibilidad.

Parámetros:

- **`--db-subnet-group-name`**: obligatorio. Nombre de este grupo de subredes.
- **`--db-subnet-group-description`**: obligatorio. Descripción de este grupo de subredes.
- **`--subnet-ids`**: obligatorio. Lista de las subredes que se van a incluir en este grupo de subredes. Ejemplo: `subnet-53ab3636`.
- –Etiquetas: opcional. Lista de las etiquetas (pares clave-valor) que se van a asociar a este grupo de subredes.

El código siguiente crea el grupo de subredes `sample-subnet-group` con tres subredes, `subnet-4e26d263`, `subnet-afc329f4` y `subnet-b3806e8f`.

Para 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
```

Para 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
```

La salida de esta operación será similar a lo que se indica a continuación (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"
    }
}
```

Descripción de un grupo de subredes de Amazon DocumentDB

Puede usar el Consola de administración de AWS o el AWS CLI para obtener los detalles de un grupo de subredes de Amazon DocumentDB.

Using the Consola de administración de AWS

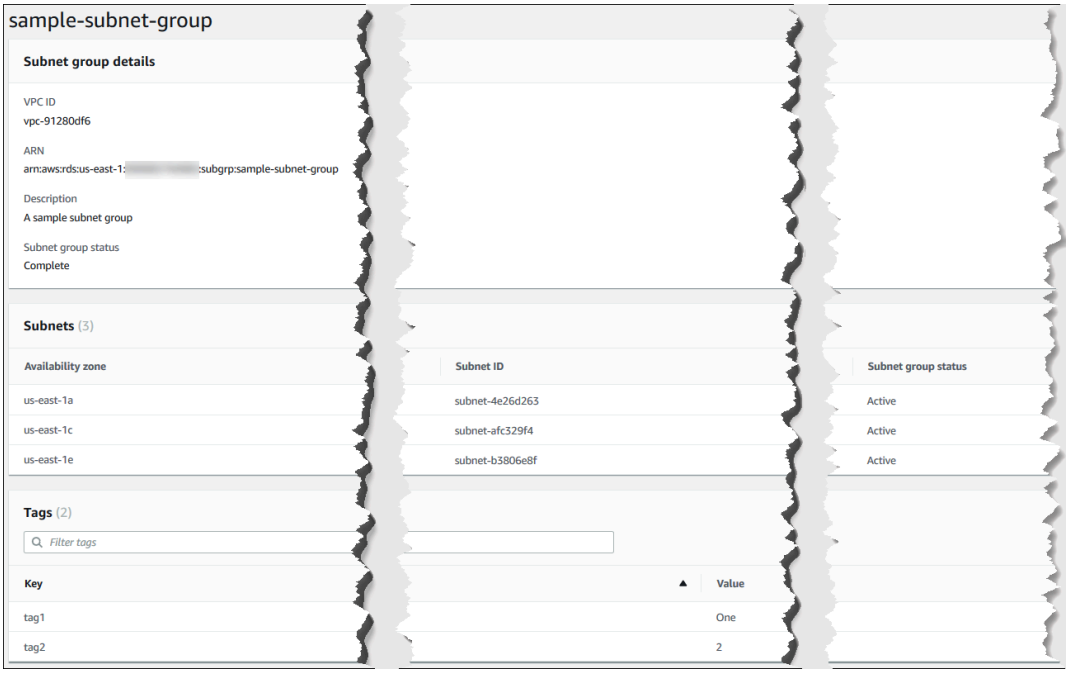
En el siguiente procedimiento se muestra cómo obtener detalles de un grupo de subredes de Amazon DocumentDB.

Para obtener los detalles de un grupo de subredes

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, elija Subnet groups (grupos de subredes).

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. Para consultar los detalles de un grupo de subredes seleccione el nombre de dicho grupo.



Using the AWS CLI

Para obtener los detalles de un grupo de subredes de Amazon DocumentDB, utilice la operación `describe-db-subnet-groups` con el siguiente parámetro.

Parámetro

- `--db-subnet=group-name`: opcional. Si se incluye, se muestran los detalles del grupo de subredes nombrado. Si se omite, se muestran los detalles de hasta 100 grupos de subredes.

Example

El siguiente código muestra los detalles del grupo de subredes `sample-subnet-group` que creamos en la sección Creación de un grupo de subredes de Amazon DocumentDB.

Para Linux, macOS o Unix:

```
aws docdb describe-db-subnet-groups \
    --db-subnet-group-name sample-subnet-group
```

Para Windows:

```
aws docdb describe-db-subnet-groups ^
```

```
    --db-subnet-group-name sample-subnet-group
```

La salida de esta operación será similar a lo que se indica a continuación (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"
    }
}
```

Modificación de un grupo de subredes de Amazon DocumentDB

Puede usar el Consola de administración de AWS o AWS CLI para modificar la descripción de un grupo de subredes o para agregar o eliminar subredes de un grupo de subredes de Amazon DocumentDB. Sin embargo, no puede modificar el grupo de subredes `default`.

Using the Consola de administración de AWS

Puede utilizarla Consola de administración de AWS para cambiar la descripción de un grupo de subredes o para agregar y eliminar subredes. Recuerde que, cuando termine, debe tener al menos dos zonas de disponibilidad asociadas al grupo de subredes.

Para modificar un grupo de subredes

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, elija Subnet groups (grupos de subredes). A continuación, elija el botón situado a la izquierda del nombre del grupo de subredes. Recuerde que no puede modificar el grupo de subredes `default`.

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. Elija Actions (Acciones) y después Modify (Modificar).
4. Descripción: para cambiar la descripción del grupo de subredes, introduzca una descripción nueva.
5. Para cambiar las subredes asociadas a un grupo de subredes, realice uno o varios de los siguientes pasos en la sección Add subnets (Añadir subredes):
   - Para eliminar todas las subredes de este grupo de subredes, seleccione Remove all (Eliminar todo).
   - Para eliminar subredes específicas de este grupo de subredes, seleccione Remove (Eliminar) en cada subred que desee eliminar.
   - Para añadir todas las subredes asociadas a esta VPC, seleccione Add all the subnets related to this VPC (Añadir todas las subredes asociadas a esta VPC).

- Para añadir subredes específicas a este grupo de subredes, realice los siguientes pasos para cada zona de disponibilidad para la que desee añadir una subred.

  a. Zona de disponibilidad: de la lista, elija una nueva zona de disponibilidad.

  b. Subred: de la lista, elija una subred de la zona de disponibilidad elegida para este grupo de subredes.

  c. Seleccione Add subnet (Añadir subred).

6. En el cuadro de diálogo de confirmación:
   - Para hacer estos cambios en el grupo de subredes, elija Modify (Modificar).
   - Para que el grupo de subredes permanezca sin modificaciones, seleccione Cancel (Cancelar).

Using the AWS CLI

Puede utilizarla AWS CLI para cambiar la descripción de un grupo de subredes o para añadir y eliminar subredes. Recuerde que, cuando termine, debe tener al menos dos zonas de disponibilidad asociadas al grupo de subredes. No se puede modificar el grupo de subredes `default`.

Parámetros:

- `--db-subnet-group-name`: obligatorio. Nombre del grupo de subredes de Amazon DocumentDB que está modificando.
- `--subnet-ids`: obligatorio. Lista de las subredes que debe haber en el grupo de subredes después de realizar este cambio.

  > **Important**
  >
  > Cualquier subred que actualmente se encuentre en el grupo de subredes y que no se incluya en esta lista se eliminará del grupo de subredes. Si desea mantener cualquiera de las subredes que actualmente se encuentra en el grupo de subredes, debe incluirla en esta lista.

- `--db-subnet-group-description`: opcional. Descripción del grupo de subredes.

Example

El siguiente código modifica la descripción y sustituye las subredes existentes por las subredes `subnet-991cb8d0`, `subnet-53ab3636` y `subnet-29ab1025`.

Para 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"
```

Para 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"
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON). Tenga en cuenta que este es el mismo grupo de subredes que se creó en la sección Creación de un grupo de subredes de Amazon DocumentDB. Sin embargo, las subredes del grupo de subredes se sustituyen por las que se muestran en la operación `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"
    }
}
```

Eliminación de un grupo de subredes de Amazon DocumentDB

Puede usar Consola de administración de AWS o eliminar un grupo de AWS CLI subredes de Amazon DocumentDB. Sin embargo, no puede eliminar el grupo de subredes `default`.

Using the Consola de administración de AWS

Puede usar el Consola de administración de AWS para eliminar un grupo de subredes. Sin embargo, no se puede eliminar el grupo de subredes `default`.

Para eliminar un grupo de subredes

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, elija Subnet groups (grupos de subredes). A continuación, elija el botón situado a la izquierda del nombre del grupo de subredes. Recuerde que no puede eliminar el grupo de subredes `default`.

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. Elija Acciones y, a continuación, elija Eliminar.
4. En el cuadro de diálogo de confirmación:
   - Para eliminar el grupo de subredes, seleccione Delete (Eliminar).
   - Para mantener el grupo de subredes, seleccione Cancel (Cancelar).

Using the AWS CLI

Para eliminar un grupo de subredes de Amazon DocumentDB mediante AWS CLI, utilice la `delete-db-subnet-group` operación con el siguiente parámetro.

Parámetro

- `--db-subnet-group-name`: obligatorio. Nombre del grupo de subredes de Amazon DocumentDB que se va a eliminar. Recuerde que no puede eliminar el grupo de subredes `default`.

Example

El código siguiente elimina `sample-subnet-group`.

Para Linux, macOS o Unix:

```
aws docdb delete-db-subnet-group \
    --db-subnet-group-name sample-subnet-group
```

Para Windows:

```
aws docdb delete-db-subnet-group ^
    --db-subnet-group-name sample-subnet-group
```

Esta operación no produce ningún resultado.

Alta disponibilidad y replicación de Amazon DocumentDB

Puede conseguir un alto nivel de disponibilidad y escalado de lectura en Amazon DocumentDB (compatible con MongoDB) mediante instancias de réplica. Un único clúster de Amazon DocumentDB admite una sola instancia principal y hasta 15 instancias de réplica. Dichas instancias

pueden distribuirse entre las zonas de disponibilidad dentro de la región del clúster. La instancia principal acepta el tráfico de lectura y escritura, y las instancias de réplica solo aceptan solicitudes de lectura.

El volumen del clúster consta de varias copias de los datos del clúster. Sin embargo, los datos del volumen del clúster se representan como un único volumen lógico para la instancia principal y para las réplicas de Amazon DocumentDB del clúster. Las instancias de réplica presentan consistencia final. Devuelven los resultados de las consultas con un retardo de réplica mínimo, normalmente muy inferior a 100 milisegundos una vez que la instancia principal ha escrito una actualización. El retardo de la réplica varía en función de la velocidad de cambio de la base de datos. Es decir, durante los periodos en los que se produce una gran cantidad de operaciones de escritura en la base de datos, puede registrarse un aumento del retardo de réplica.

Escalado de lectura

Las réplicas de Amazon DocumentDB funcionan bien para el escalado de lectura porque están totalmente dedicadas a las operaciones de lectura en el volumen del clúster. Las operaciones de escritura se administran en la instancia principal. El volumen del clúster lo comparten todas las instancias del clúster. Por lo tanto, no tiene que replicar y mantener una copia de los datos de cada réplica de Amazon DocumentDB.

Alta disponibilidad

Cuando se crea un clúster de Amazon DocumentDB, dependiendo del número de zonas de disponibilidad del grupo de subredes (debe haber al menos dos), Amazon DocumentDB aprovisiona instancias en las distintas zonas de disponibilidad. Cuando se crea las instancias del clúster, Amazon DocumentDB distribuye automáticamente las instancias entre las zonas de disponibilidad de un grupo de subredes para equilibrar el clúster. Esta acción también impide que todas las instancias se encuentren en la misma zona de disponibilidad.

Ejemplo

Para ilustrarlo, imagine un ejemplo en el que se crea un clúster que tiene un grupo de subredes con tres zonas de disponibilidad: AZ1, AZ2 y AZ3.

Cuando se crea la primera instancia del clúster, es la instancia principal y se sitúa en una de las zonas de disponibilidad. En este ejemplo, está en AZ1. La segunda instancia que se crea es una instancia de réplica y se sitúa en una de las otras dos zonas de disponibilidad, por ejemplo AZ2. La tercera instancia que se crea también es una instancia de réplica y se sitúa en la zona de

disponibilidad restante, AZ3. Si se crean más instancias, estas se distribuyen entre las distintas zonas de disponibilidad para lograr el equilibrio en el clúster.

Si se produce un error en la instancia principal (AZ1), se activará una conmutación por error y una de las réplicas existentes pasará a ser la instancia principal. Cuando se recupere la antigua instancia principal, se convertirá en una réplica en la misma zona de disponibilidad en la que se aprovisionó (AZ1). Al suministrar un clúster de tres instancias, Amazon DocumentDB sigue conservando ese clúster de tres instancias. Amazon DocumentDB gestiona automáticamente la detección, la conmutación por error y la recuperación de los errores de las instancias sin ninguna intervención manual.

Cuando Amazon DocumentDB realiza una conmutación por error y recupera una instancia, la instancia recuperada permanece en la zona de disponibilidad en la que se aprovisionó originalmente. Sin embargo, el rol de la instancia podría cambiar de principal a réplica. Esto tiene por objeto evitar que una serie de conmutaciones por error pueda provocar que todas las instancias se encuentren en la misma zona de disponibilidad.

Puede especificar réplicas de Amazon DocumentDB para que actúen como destino para las conmutaciones por error. Es decir, si la instancia principal falla, la réplica de un tercero o la réplica de Amazon DocumentDB que ha especificado pasa a ser la instancia principal. En este proceso se produce una breve interrupción durante la cual las solicitudes de escritura y lectura realizadas a la instancia principal generan errores con una excepción. Si el clúster de Amazon DocumentDB no incluye ninguna réplica de Amazon DocumentDB; la instancia principal se vuelve cuando produce un error. Promover una réplica de Amazon DocumentDB es mucho más rápido que volver a crear la instancia principal.

Para lograr una alta disponibilidad, cree una o más réplicas de Amazon DocumentDB de la misma clase de instancia que la instancia principal en diferentes zonas de disponibilidad.

Para obtener más información, consulte los siguientes temas:

- Información general de la tolerancia a errores de clúster de Amazon DocumentDB
- Conmutación por error de Amazon DocumentDB
  - Control del destino de la conmutación por error

En las versiones 5.0 y 8.0 de Amazon DocumentDB, las instancias de réplica no se reinician cuando se reinicia la instancia principal. Siguen atendiendo las solicitudes de lectura durante los reinicios de la instancia principal, lo que mantiene una mejor disponibilidad.

Alta disponibilidad con clústeres globales

Para lograr una alta disponibilidad en varios clústeres Regiones de AWS, puede configurar clústeres globales de Amazon DocumentDB. Cada clúster global abarca varias regiones, lo que permite realizar lecturas globales de baja latencia y recuperar ante desastres en caso de interrupciones en distintas regiones. Amazon DocumentDB gestiona automáticamente la replicación de todos los datos y actualizaciones de la región principal a cada una de las regiones secundarias.

Adición de réplicas de

La primera instancia que se añade al clúster es la instancia principal. Cada instancia de base de datos que se añade después de la primera instancia es una instancia de réplica. Un clúster puede tener hasta 15 instancias de réplica, además de la instancia principal.

Al crear un clúster con el Consola de administración de AWS, se crea automáticamente una instancia principal al mismo tiempo. Para crear una réplica al mismo tiempo que crea el clúster y la instancia principal, seleccione Create replica in different zone (Crear réplica en zona diferente). Para obtener más información, consulte el paso 4.d de Creación de un clúster de Amazon DocumentDB. Para añadir más réplicas a un clúster de Amazon DocumentDB, consulte Agregación de una instancia de Amazon DocumentDB a un clúster.

Cuando utilices el AWS CLI para crear tu clúster, debes crear de forma explícita tus instancias principales y de réplica. Para obtener más información, consulta la sección «Uso de AWS CLI» en los siguientes temas:

- Creación de un clúster de Amazon DocumentDB
- Agregación de una instancia de Amazon DocumentDB a un clúster

Conmutación por error de Amazon DocumentDB

En algunos casos, como en determinados tipos de mantenimiento planificado, o en el improbable caso de que se produzca un error en el nodo principal o en la zona de disponibilidad, Amazon DocumentDB (compatible con MongoDB) detectará el error y reemplazará el nodo principal. Durante una conmutación por error, el tiempo de inactividad de escritura se minimiza. Esto se debe a que el papel del nodo principal se conmuta por error a una de las réplicas de lectura en lugar de tener que crear y aprovisionar un nuevo nodo principal. Esta detección de error y promoción de réplica garantizan la posibilidad de reanudar la escritura en el nuevo principal tan pronto como se complete la promoción.

Para que la conmutación por error funcione, el clúster debe tener como mínimo dos instancias: una instancia principal y al menos una de réplica.

> **Note**
>
> Este tema solo se aplica a los clústeres basados en instancias de Amazon DocumentDB. No se aplica a los clústeres elásticos o globales.

Control del destino de la conmutación por error

Amazon DocumentDB proporciona niveles de conmutación por error como una forma de controlar qué instancia de réplica pasa a ser la instancia principal cuando se produce una conmutación por error.

Niveles de conmutación por error

Cada instancia de réplica está asociada a un nivel de conmutación por error (0-15). Cuando se produce una conmutación por error debido a tareas de mantenimiento o al caso improbable de que se produzca un error de hardware, la instancia principal se conmuta a una réplica con el nivel de prioridad mayor (el nivel más bajo). Si varias réplicas tienen el mismo nivel de prioridad, la instancia principal se conmutará a la réplica de ese nivel cuyo tamaño sea lo más similar a la principal anterior.

Estableciendo el nivel de conmutación por error para un grupo de réplicas seleccionadas en `0` (la prioridad más alta), puede asegurarse de que una conmutación por error promoverá una de las réplicas de ese grupo. En la práctica, puede evitar determinadas réplicas pasen a ser la instancia principal en caso de que se produzca una conmutación por error asignando un nivel de prioridad bajo (un número alto) a esas réplicas. Esto resulta útil en aquellos casos en los que determinadas réplicas se usan ampliamente en una aplicación y la conmutación por error a una de ellas afectaría negativamente a una aplicación crítica.

Se recomienda que las réplicas que son posibles objetivos de conmutación por error utilicen la misma clase de instancia que la instancia principal. Esto evita la degradación del rendimiento de escritura si una conmutación por error promueve la creación de una réplica durante las actualizaciones del sistema operativo u otros tipos de mantenimiento.

Puede configurar el nivel de conmutación por error de una instancia cuando la cree o modificándola más adelante. La configuración del nivel conmutación por error de una instancia modificando la instancia no genera una conmutación por error. Para obtener más información, consulte los siguientes temas:

- Agregación de una instancia de Amazon DocumentDB a un clúster
- Modificación de una instancia de base de datos de Amazon DocumentDB

Al iniciar manualmente una conmutación por error, dispone de dos métodos para controlar qué instancia de réplica pasa a ser la principal: los niveles de conmutación por error indicados anteriormente y el parámetro `--target-db-instance-identifier`.

-- identificador de instancia de base de datos de destino

Para las pruebas, puede forzar una conmutación por error mediante la operación `failover-db-cluster`. Puede utilizar el parámetro `--target-db-instance-identifier` para especificar qué réplica pasará a ser la instancia principal. El uso del parámetro `--target-db-instance-identifier` invalida el nivel de prioridad de conmutación por error. Si no especifica el parámetro `--target-db-instance-identifier`, la conmutación por error de la instancia principal se rige por el nivel de prioridad de conmutación por error.

¿Qué ocurre durante una conmutación por error?

Amazon DocumentDB administra automáticamente la conmutación por error para que sus aplicaciones puedan reanudar las operaciones de la base de datos lo antes posible y sin intervención administrativa.

- Si dispone de una réplica de Amazon DocumentDB en la misma zona de disponibilidad o en otra distinta cuando se realiza la conmutación por error, Amazon DocumentDB cambia el registro de nombre canónico (CNAME) para que su instancia apunte a la réplica en buen estado que, a su vez, se convierte en la nueva instancia principal. La conmutación por error suele completarse en 30 segundos de principio a fin.
- Si no tiene una instancia de réplica de Amazon DocumentDB (por ejemplo, un clúster de instancia única), Amazon DocumentDB intentará crear una nueva instancia en la misma zona de disponibilidad que la instancia original. Este reemplazo de la instancia original se lleva a cabo con el mayor esfuerzo, pero puede fallar si, por ejemplo, existe un problema que esté afectando a la zona de disponibilidad de manera generalizada.

La aplicación debe reintentar establecer las conexiones de la base de datos en caso de que se pierda la conexión.

Prueba de conmutación por error

Una conmutación por error de un clúster promueve una de las réplicas de Amazon DocumentDB (instancias de solo lectura) del clúster a instancia principal (la instancia de escritura del clúster).

Cuando se produce un error en la instancia principal, Amazon DocumentDB conmuta automáticamente a una réplica de Amazon DocumentDB, si existe. Puede forzar una conmutación por error cuando desee simular un error en una instancia principal para realizar pruebas. Cada instancia de un clúster tiene su propia dirección de punto de conexión. Por lo tanto, es necesario eliminar y restablecer las conexiones existentes que utilizan dichas direcciones de punto de conexión cuando finalice la conmutación por error.

Para forzar una conmutación por error, utilice la operación `failover-db-cluster` con los parámetros que se indican a continuación.

- `--db-cluster-identifier`: obligatorio. El nombre del clúster de base de datos que se va a conmutar por error.
- `--target-db-instance-identifier`: opcional. El nombre de la instancia que pasará a ser la instancia principal.

Example

La siguiente operación fuerza una conmutación por error del clúster `sample-cluster`. No especifica qué instancia se convertirá en la nueva instancia principal, por lo que Amazon DocumentDB elige la instancia de acuerdo con el nivel de prioridad de conmutación por error.

Para Linux, macOS o Unix:

```
aws docdb failover-db-cluster \
   --db-cluster-identifier sample-cluster
```

Para Windows:

```
aws docdb failover-db-cluster ^
   --db-cluster-identifier sample-cluster
```

La siguiente operación fuerza una conmutación por error del clúster `sample-cluster`, especificando que `sample-cluster-instance` pasará a ser la instancia principal. (Observe `"IsClusterWriter": true` en el resultado).

Para Linux, macOS o Unix:

```
aws docdb failover-db-cluster \
   --db-cluster-identifier sample-cluster \
   --target-db-instance-identifier sample-cluster-instance
```

Para Windows:

```
aws docdb failover-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --target-db-instance-identifier sample-cluster-instance
```

La salida de esta operación será similar a lo que se indica a continuación (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"
    }
}
```

Retraso de replicación

El retraso de replicación suele ser de 50 ms o menos. Los motivos más comunes del aumento del retraso en la réplica son:

- Una velocidad de escritura alta en la principal que hace que las réplicas de lectura queden por detrás de la principal.
- Discrepancia en las réplicas de lectura entre consultas de larga duración (p. ej., escaneos secuenciales de gran tamaño o consultas de agregación) y la replicación de escritura entrante.

- Gran cantidad de consultas simultáneas en las réplicas de lectura.

Para minimizar el retraso en la replicación, pruebe estas técnicas de solución de problemas:

- Si tienes una velocidad de escritura alta o un uso elevado de la CPU, amplía las instancias de tu clúster.
- Si las réplicas leídas contienen consultas de larga duración y se actualizan con mucha frecuencia los documentos consultados, plantéate la posibilidad de modificar las consultas de larga duración o de ejecutarlas en la primary/write réplica para evitar problemas en las réplicas leídas.
- Si hay un gran número de consultas simultáneas o un uso elevado de la CPU solo en las réplicas de lectura, otra opción es escalar horizontalmente el número de réplicas de lectura para distribuir la carga de trabajo.
- Dado que el retraso en la replicación se debe a un alto rendimiento de escritura y a consultas de ejecución prolongada, solucione el retraso de la replicación utilizando la `DBClusterReplicaLagMaximum` CloudWatch métrica en combinación con las métricas /y del registrador de consultas lento. `WriteThroughput` `WriteIOPS`

Asegúrese de que todas las réplicas sean del mismo tipo de instancia para que la conmutación por error de un clúster no reduzca el rendimiento.

Si va a elegir entre escalar verticalmente y horizontalmente (por ejemplo, seis instancias más pequeñas frente a tres instancias más grandes), normalmente recomendamos que primero intente escalar verticalmente (instancias más grandes) antes de hacerlo horizontalmente, ya que obtendrá una caché de búfer más grande por instancia de base de datos.

De forma proactiva, debe configurar una alarma de retraso de replicación y establecer su umbral en un valor que considere que es el límite superior del retraso (u “obsoleto”) que pueden estar los datos de las instancias de réplica antes de que comiencen a afectar a la funcionalidad de la aplicación. En general, le recomendamos que se supere el umbral de retraso de replicación en varios puntos de datos antes de emitir una alarma debido a las cargas de trabajo transitorias.

> **Note**
>
> Configure otra alarma para los retrasos de replicación que superen los 10 segundos. Si supera este umbral para varios puntos de datos, amplíe sus instancias o reduzca el rendimiento de escritura en la instancia principal.

Administración de eventos de Amazon DocumentDB

Amazon DocumentDB (compatible con MongoDB) mantiene un registro de los eventos relacionados con los clústeres, las instancias, las instantáneas, los grupos de seguridad y los grupos de parámetros de clúster. Esta información incluye la fecha y hora del evento, el nombre del origen y el tipo del origen del evento y un mensaje relacionado con el evento.

> **Important**
>
> En determinadas características de administración, Amazon DocumentDB utiliza una tecnología operativa que comparte con Amazon RDS y Amazon Neptune. Los límites regionales, es decir, los límites que se aplican en el nivel de región, se comparten entre Amazon DocumentDB, Amazon RDS, y Amazon Neptune. Para obtener más información, consulte Cuotas regionales.

Temas

- Visualización de las categorías de eventos de Amazon DocumentDB
- Visualización de eventos de Amazon DocumentDB

Visualización de las categorías de eventos de Amazon DocumentDB

Cada tipo de recurso de Amazon DocumentDB, tiene tipos específicos de eventos que se pueden asociar a él. Puede utilizar la AWS CLI `describe-event-categories` operación para ver el mapeo entre los tipos de eventos y los tipos de recursos de Amazon DocumentDB.

Parameters

- **`--source-type`**: opcional. Utilice el parámetro `--source-type` para ver las categorías de eventos de un tipo de origen particular. Se permiten los siguientes valores:
  - `db-cluster`
  - `db-instance`
  - `db-parameter-group`
  - `db-security-group`
  - `db-cluster-snapshot`

- **`--filters`**: opcional. Para ver categorías de eventos únicamente para Amazon DocumentDB, utilice el filtro `--filter Name=engine,Values=docdb`.

Example

El siguiente código muestra las categorías de eventos asociadas a los clústeres.

Para Linux, macOS o Unix:

```
aws docdb describe-event-categories \
    --filter Name=engine,Values=docdb \
    --source-type db-cluster
```

Para Windows:

```
aws docdb describe-event-categories ^
    --filter Name=engine,Values=docdb ^
    --source-type db-cluster
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
{
    "EventCategoriesMapList": [
        {
            "EventCategories": [
                "notification",
                "failure",
                "maintenance",
                "failover"
            ],
            "SourceType": "db-cluster"
        }
    ]
}
```

El siguiente código muestra las categorías de eventos asociadas a cada tipo de recurso de Amazon DocumentDB.

```
aws docdb describe-event-categories
```

La salida de esta operación será similar a lo que se indica a continuación (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"
            ]
        }
    ]
}
```

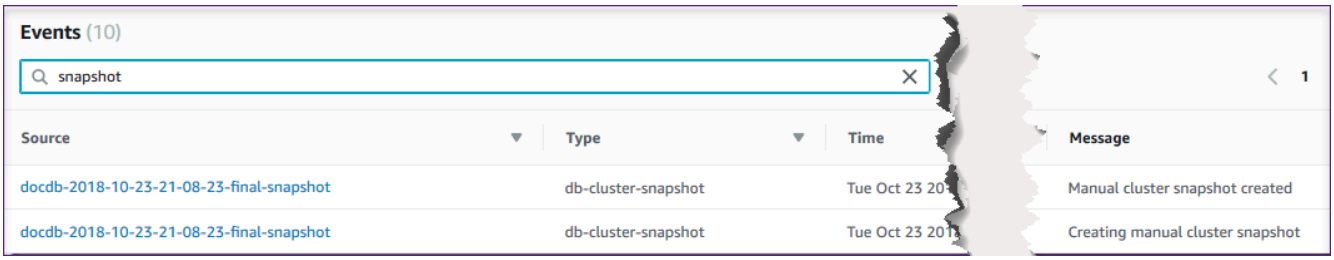
Visualización de eventos de Amazon DocumentDB

Puede recuperar eventos de los recursos de Amazon DocumentDB a través de la consola de Amazon DocumentDB, la cual muestra los eventos de las últimas 24 horas. También puede recuperar eventos para sus recursos de Amazon DocumentDB mediante el AWS CLI comando describe-events o la operación de la API Amazon DescribeEventsDocumentDB. Si utiliza la API AWS CLI o la API de Amazon DocumentDB para ver los eventos, puede recuperar los eventos de los últimos 14 días.

Using the Consola de administración de AWS

Visualización de todos los eventos de las instancias de Amazon DocumentDB de las últimas 24 horas

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Events. Los eventos disponibles aparecen en una lista.
3. Utilice la lista Filter (Filtro) para filtrar los eventos por tipo. Escriba un término en el cuadro de texto para filtrar aún más los resultados. Por ejemplo, la siguiente captura de pantalla muestra la filtración de todos los eventos de Amazon DocumentDB para los eventos de instantánea.



Using the AWS CLI

Visualización de todos los eventos de las instancias de Amazon DocumentDB de los últimos 7 días

Puede ver todos los eventos de la instancia de Amazon DocumentDB de los últimos 7 días ejecutando la AWS CLI operación describe-events con el `--duration` parámetro establecido en (10 080 minutos). `10080`

```
aws docdb describe-events --duration 10080
```

Filtrado de eventos de Amazon DocumentDB

Para ver eventos de Amazon DocumentDB específicos, utilice la operación `describe-events` con los siguientes parámetros.

Parameters

- **`--filter`**: necesario para limitar los valores devueltos a los eventos de Amazon DocumentDB. Utilice **`Name=engine,Values=docdb`** para filtrar todos los eventos únicamente para Amazon DocumentDB.
- **`--source-identifier`**: opcional. El identificador del origen de eventos para el que se devuelven eventos. Si se omite, se incluyen en los resultados los eventos de todos los orígenes.
- **`--source-type`**: opcional, a menos que se proporcione `--source-identifier`, entonces es obligatorio. Si se proporciona `--source-identifier`, `--source-type` debe coincidir con el tipo de `--source-identifier`. Se permiten los siguientes valores:
  - `db-cluster`
  - `db-instance`
  - `db-parameter-group`
  - `db-security-group`
  - `db-cluster-snapshot`

El siguiente ejemplo muestra todos los eventos de Amazon DocumentDB.

```
aws docdb describe-events --filters Name=engine,Values=docdb
```

La salida de esta operación será similar a lo que se indica a continuación (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"
            ]
        }
    ]
}
```

Para obtener más información, consulte Auditoría de eventos de Amazon DocumentDB.

Administración de los grupos de parámetros de clúster de Amazon DocumentDB

Es posible administrar la configuración del motor de Amazon DocumentDB mediante los parámetros de un grupo de parámetros de clúster. Un grupo de parámetros de clúster es una colección de valores de configuración de Amazon DocumentDB que facilitan la administración de los parámetros de los clústeres de Amazon DocumentDB. Los grupos de parámetros de clúster actúan como un contenedor para los valores de configuración del motor que se aplican a todas las instancias del clúster.

En esta sección se describe cómo crear, ver y modificar grupos de parámetros de clúster. También muestra cómo se puede determinar qué grupo de parámetros de clúster está asociado a un clúster determinado.

Temas

- Descripción de los grupos de parámetros de clúster de Amazon DocumentDB
- Creación de grupos de parámetros de clúster de Amazon DocumentDB
- Modificación de grupos de parámetros de clúster de Amazon DocumentDB
- Modificación de clústeres de Amazon DocumentDB para utilizar grupos de parámetros de clúster personalizados
- Copia de grupos de parámetros de clúster de Amazon DocumentDB
- Restablecimiento de grupos de parámetros de clúster de Amazon DocumentDB
- Borrado de grupos de parámetros de clúster de Amazon DocumentDB
- Referencia de parámetros de clúster de Amazon DocumentDB

Descripción de los grupos de parámetros de clúster de Amazon DocumentDB

Un grupo de parámetros de `default` clúster se crea automáticamente cuando se crea el primer clúster de Amazon DocumentDB en un motor nuevo Región de AWS o se utiliza uno nuevo. Los clústeres subsiguientes, que se crean en la misma región y tienen la misma versión de motor, se crean con el grupo de parámetros del `default` clúster.

Temas

- Descripción de los detalles de un grupo de parámetros de clúster de Amazon DocumentDB
- Determinación del grupo de parámetros de un clúster de Amazon DocumentDB
- Determinar los clústeres e instancias asociados a un grupo de parámetros de clústeres de Amazon DocumentDB

Descripción de los detalles de un grupo de parámetros de clúster de Amazon DocumentDB

Para describir los detalles de un grupo de parámetros de clúster determinado, complete los pasos siguientes con Consola de administración de AWS o AWS Command Line Interface (AWS CLI).

Using the Consola de administración de AWS

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Grupos de parámetros.

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En el panel Cluster parameter groups (Grupos de parámetros de clúster), seleccione el nombre del grupo de parámetros cuyos detalles desee ver.
4. La página resultante muestra los parámetros del grupo de parámetros, la actividad reciente y las etiquetas.

   - En Cluster parameters (Parámetros de clúster), puede ver el nombre del parámetro, el valor actual, los valores permitidos, si el parámetro es modificable, su tipo de aplicación, tipo de datos y descripción. Puede modificar parámetros individuales seleccionando el parámetro y eligiendo Edit (Editar) en la sección Cluster parameters (Parámetros de clúster) . Para obtener más información, consulte Modificación de parámetros de clúster de Amazon DocumentDB.
   - En Recent events (Eventos recientes), puede ver los eventos más recientes de este grupo de parámetros. Puede filtrar estos eventos mediante la barra de búsqueda de esta

sección. Para obtener más información, consulte Administración de eventos de Amazon DocumentDB.

- En Tags (Etiquetas), puede ver las etiquetas que se encuentran en este grupo de parámetros de clúster. Puede agregar o quitar etiquetas seleccionando Edit (Editar) en la sección Tags (Etiquetas) . Para obtener más información, consulte Etiquetado de recursos de Amazon DocumentDB.

Using the AWS CLI

Puede usar el `describe-db-cluster-parameter-groups` AWS CLI comando para ver el nombre del recurso de Amazon (ARN), la familia, la descripción y el nombre de un único grupo de parámetros de clúster o de todos los grupos de parámetros de clúster que tenga para Amazon DocumentDB. También puede usar el `describe-db-cluster-parameters` AWS CLI comando para ver los parámetros y sus detalles dentro de un único grupo de parámetros de clúster.

- **`--describe-db-cluster-parameter-groups`**: para ver una lista de todos los grupos de parámetros de clúster y sus detalles.
  - **`--db-cluster-parameter-group-name`**: opcional. El nombre del grupo de parámetros de clúster que desea describir. Si se omite este parámetro, se describen todos los grupos de parámetros de clúster.
- **`--describe-db-cluster-parameters`**: para obtener una lista de los parámetros de un grupo de parámetros, junto con sus valores.
  - **`--db-cluster-parameter-group name`**: obligatorio. El nombre del grupo de parámetros de clúster que desea describir.

Example

El código siguiente enumera hasta 100 grupos de parámetros de clúster y su ARN, familia, descripción y nombre.

```
aws docdb describe-db-cluster-parameter-groups
```

La salida de esta operación será similar a lo que se indica a continuación (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

El código siguiente enumera el ARN, la familia, la descripción y el nombre de `sample-parameter-group`.

Para Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameter-groups \
        --db-cluster-parameter-group-name sample-parameter-group
```

Para Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
        --db-cluster-parameter-group-name sample-parameter-group
```

La salida de esta operación será similar a lo que se indica a continuación (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

El código siguiente enumera los valores de los parámetros de `sample-parameter-group`.

Para Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

Para Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

La salida de esta operación será similar a lo que se indica a continuación (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",
```

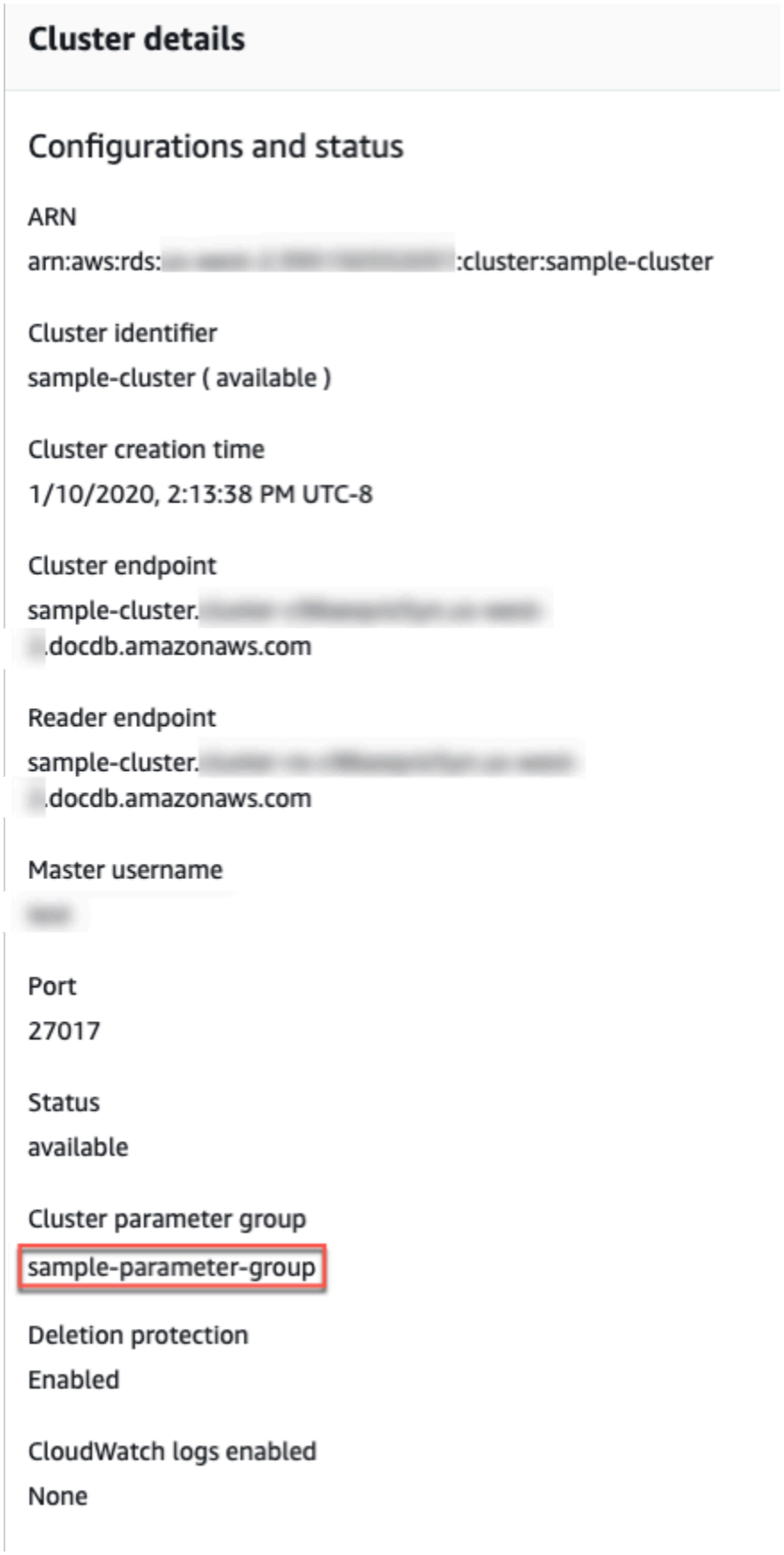
```
            "ParameterValue": "17777",
            "Description": "Duration of time in seconds that the change stream log
 is retained and can be consumed.",
            "Source": "user",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "3600-86400",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
         }
   ]
}
```

Determinación del grupo de parámetros de un clúster de Amazon DocumentDB

Para determinar qué grupo de parámetros está asociado a un clúster en particular, complete los pasos siguientes con Consola de administración de AWS o con AWS CLI.

Using the Consola de administración de AWS

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación izquierdo, elija Clusters (Clústeres).
3. En la lista de clústeres, elija el nombre del clúster de su interés.
4. La página resultante muestra los detalles del clúster seleccionado. Desplácese hacia abajo hasta Cluster details (Detalles del clúster). En la parte inferior de esa sección, busque el nombre del grupo de parámetros debajo del grupo de parámetros de clúster.


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

El AWS CLI código siguiente determina qué grupo de parámetros gobierna el clústersample-cluster.

```
aws docdb describe-db-clusters \
      --db-cluster-identifier sample-cluster \
      --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
[
       [
           "sample-cluster",
           "sample-parameter-group"
       ]
]
```

Determinar los clústeres e instancias asociados a un grupo de parámetros de clústeres de Amazon DocumentDB

Para determinar cuántos clústeres e instancias están asociados a cada grupo de parámetros, complete los pasos siguientes mediante el AWS CLI.

Using the AWS CLI

El AWS CLI código siguiente determina a cuántos clústeres e instancias están asociadossample-parameter-group.

Para 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
      }) | .[]'
```

Para 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
      }) | .[]'
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
{
   "parameter_group_name": "sample-parameter-group",
   "total_clusters": 50,
   "total_instances": 150
}
```

Creación de grupos de parámetros de clúster de Amazon DocumentDB

Los grupos de parámetros de clúster predeterminados `default.docdb5.0default.docdb4.0`, como, o`default.docdb3.6`,, se crean al crear un clúster con una nueva versión del motor y en una nueva Región de AWS. Los clústeres posteriores creados en la misma región y con la misma versión de motor heredan el grupo de parámetros del `default` clúster. Una vez creados, los grupos de parámetros `default` no se pueden eliminar ni cambiar de nombre. Puede modificar el comportamiento del motor de las instancias de clúster creando un grupo de parámetros personalizado con los valores de parámetros preferidos y adjuntándolo a su clúster de Amazon DocumentDB.

El siguiente procedimiento le guía a través de la creación de un grupo de parámetros de clúster personalizado. A continuación, puede modificar los parámetros dentro de ese grupo de parámetros.

> **Note**
>
> Después de crear un grupo de parámetros de clúster, debe esperar al menos 5 minutos antes de usar ese grupo de parámetros en concreto. Esto permite a `create` finalizar por

completo la acción Amazon DocumentDB antes de que el grupo de parámetros de clúster se use para un nuevo clúster. Puede utilizar la operación Consola de administración de AWS o la `describe-db-cluster-parameter-groups` AWS CLI operación para comprobar que se ha creado el grupo de parámetros del clúster. Para obtener más información, consulte Descripción de los grupos de parámetros de clúster de Amazon DocumentDB.

Using the Consola de administración de AWS

Creación de un grupo de parámetros de clúster

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Grupos de parámetros.

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En el panel Cluster parameter groups (Grupos de parámetros del clúster), elija Create (Crear).
4. En el panel Create cluster parameter group (Crear grupo de parámetros de clúster) escriba lo siguiente:

   a. Nombre del nuevo grupo de parámetros de clúster: escriba un nombre para el grupo de parámetros de clúster. Por ejemplo, `sample-parameter-group`. Los grupos de parámetros de clúster tienen las siguientes restricciones de denominación:

      - La longitud es de [1-255] caracteres alfanuméricos.
      - El primer carácter debe ser una letra.
      - No puede terminar con un guion ni contener dos guiones consecutivos.

   b. Familia: elija la versión de DocumentDB que desee usar para su clúster.

   c. Descripción: proporcione una descripción para este grupo de parámetros de clúster.

5. Para crear el grupo de parámetros de clúster, elija Create (Crear). Para cancelar la operación, elija Cancel (Cancelar).

6. Después de seleccionar Create (Crear), aparece el siguiente texto en la parte superior de la página para comprobar que el grupo de parámetros de clúster se ha creado correctamente:

```
Successfully created cluster parameter group 'sample-parameter-group'.
```

Using the AWS CLI

Para crear un nuevo grupo de parámetros de clústeres para los clústeres de Amazon DocumentDB 4.0, utilice la AWS CLI `create-db-cluster-parameter-group` operación con los siguientes parámetros:

- **`--db-cluster-parameter-group-name`**: el nombre del grupo de parámetros de clúster de base de datos. Por ejemplo, `sample-parameter-group`.
- **`--db-cluster-parameter-group-family`**: la familia de grupos de parámetros de clúster que se utiliza como plantilla para el grupo de parámetros de clúster personalizado.
- **`--description`**: la descripción proporcionada por el usuario para este grupo de parámetros de clúster. El siguiente ejemplo utiliza "`Custom docdb4.0 parameter group`".

Para 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"
```

Para 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"
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
{
```

```
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "sample-parameter-group",
        "DBParameterGroupFamily": "docdb5.0",
        "Description": "Custom docdb4.0 parameter group",
        "DBClusterParameterGroupArn": "sample-parameter-group-arn"
    }
}
```

Modificación de grupos de parámetros de clúster de Amazon DocumentDB

En esta sección se explica cómo modificar un grupo de parámetros personalizados de Amazon DocumentDB. En Amazon DocumentDB, no puede modificar un grupo de parámetros de `default` clúster que se crea al crear por primera vez un clúster con una nueva versión del motor en una nueva región. Si el clúster de Amazon DocumentDB utiliza el grupo de parámetros de clúster predeterminado y desea modificar un valor en él, primero debe crear un nuevo grupo de parámetros o copiar un grupo de parámetros existente, modificarlo y, a continuación, aplicar el grupo de parámetros modificado a su clúster.

Siga los pasos siguientes para modificar un grupo de parámetros de clúster. Las acciones de modificación pueden tardar un tiempo en propagarse. Espere a que el grupo de parámetros del clúster modificado esté disponible antes de adjuntarlo a su clúster. Puede usar la operación Consola de administración de AWS o la AWS CLI `describe-db-cluster-parameters` operación para verificar que el grupo de parámetros del clúster se haya modificado. Para obtener más información, consulte Descripción de los grupos de parámetros de clúster.

Using the Consola de administración de AWS

Siga estos pasos para modificar un grupo de parámetros personalizado de Amazon DocumentDB. No puede modificar un grupo de parámetros `default`. Si desea modificar un valor del grupo de parámetros `default`, puede copiar el grupo de parámetros del clúster por defecto, modificarlo y, a continuación, aplicar el grupo de parámetros modificado al clúster. Para obtener más información sobre la aplicación de grupos de parámetros al clúster, consulte Modificación de un clúster de Amazon DocumentDB.

Modificación de un grupo de parámetros de clúster personalizado

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.

2. En el panel de navegación del lado izquierdo de la consola, elija Parameter groups (Grupos de parámetros). En la lista de grupos de parámetros, seleccione el nombre del grupo de parámetros que desee modificar.

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. Para cada parámetro del grupo de parámetros que desee modificar, haga lo siguiente:

   a. Localice el parámetro que desea modificar y compruebe que es modificable comprobando si aparece en la `true` columna Modifiable (Modificable) .

   b. Si es modificable, seleccione el parámetro y elija Edit (Editar) en la parte superior derecha de la página de la consola.

   c. En el cuadro de diálogo Modify (Modificar) **<parameter-name>**, realice los cambios que desee. A continuación, elija Modify cluster parameter (Modificar el parámetro de clúster), o elija Cancel (Cancelar) para descartar los cambios.

Using the AWS CLI

Puede modificar los valores de `ParameterValue`, `Description` o `ApplyMethod` de cualquier parámetro modificable de un grupo de parámetros de clúster personalizado de Amazon DocumentDB mediante la AWS CLI. No se pueden realizar modificaciones directamente en un grupo de parámetros de clúster por defecto.

Para modificar los parámetros de un grupo de parámetros de clúster personalizado, utilice la operación `modify-db-cluster-parameter-group` con los siguientes parámetros.

- **`--db-cluster-parameter-group-name`**: obligatorio. El nombre del grupo de parámetros de clúster que va a modificar.
- **`--parameters`**: obligatorio. Los parámetros que está modificando. Para obtener una lista de los parámetros que se aplican a todas las instancias de un clúster de Amazon DocumentDB, consulte Referencia de parámetros de clúster de Amazon DocumentDB. Cada entrada de parámetro debe incluir lo siguiente:
  - **`ParameterName`**: el nombre del grupo de parámetros de clúster que va a modificar.

- **`ParameterValue`**: el valor nuevo de este parámetro de clúster.
- **`ApplyMethod`**: cómo desea que se apliquen los cambios a este parámetro. Los valores permitidos son `immediate` y `pending-reboot`.

> **Note**
>
> Los parámetros con el `ApplyType` de `static` deben tener un `ApplyMethod` de `pending-reboot`.

Example- Modificación del valor de un parámetro

En este ejemplo, se muestran los valores de parámetro de `sample-parameter-group` y se modifica el parámetro `tls`. A continuación, después de 5 minutos, también se muestran los valores de parámetro de `sample-parameter-group` para ver los valores de parámetro que han cambiado.

1. Muestre los parámetros de `sample-parameter-group` y sus valores.

   Para Linux, macOS o Unix:

   ```
   aws docdb describe-db-cluster-parameters \
          --db-cluster-parameter-group-name sample-parameter-group
   ```

   Para Windows:

   ```
   aws docdb describe-db-cluster-parameters ^
          --db-cluster-parameter-group-name sample-parameter-group
   ```

   La salida de esta operación será similar a lo que se indica a continuación (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. Modifique el parámetro `tls` para que su valor sea `disabled`.

   No se puede modificar `ApplyMethod` ya que `ApplyType` es `static`.

   Para 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
```

   Para Windows:

```
aws docdb modify-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group ^
       --parameters
 "ParameterName"=tls,"ParameterValue"=disabled,"ApplyMethod"=pending-reboot
```

   La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
{
       "DBClusterParameterGroupName": "sample-parameter-group"
   }
```

3. Espere al menos 5 minutos.
4. Enumere los valores de parámetro de `sample-parameter-group` para verificar que se modificó el parámetro de `tls`.

   Para Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameters \
       --db-cluster-parameter-group-name sample-parameter-group
```

   Para Windows:

```
aws docdb describe-db-cluster-parameters ^
       --db-cluster-parameter-group-name sample-parameter-group
```

   La salida de esta operación será similar a lo que se indica a continuación (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"
           }
       ]
```

```
}
```

Modificación de clústeres de Amazon DocumentDB para utilizar grupos de parámetros de clúster personalizados

Al crear un clúster de Amazon DocumentDB, se crea automáticamente un grupo de parámetros `default.docdb4.0` para ese clúster. No se puede modificar el grupo de parámetros del clúster `default`. En su lugar, puede modificar el clúster de Amazon DocumentDB para asociar un nuevo grupo de parámetros personalizados con él.

En esta sección se explica cómo modificar un clúster de Amazon DocumentDB existente para usar un grupo de parámetros de clúster personalizado mediante el símbolo Consola de administración de AWS y el AWS Command Line Interface ()AWS CLI.

Using the Consola de administración de AWS

Modificación de un clúster de Amazon DocumentDB para utilizar un nuevo grupo de parámetros de clúster no predeterminado

1. Antes de comenzar, asegúrese de haber creado un clúster de Amazon DocumentDB y un grupo de parámetros de clúster. Consulte Creación de un clúster de Amazon DocumentDB y Creación de grupos de parámetros de clúster de Amazon DocumentDB para obtener más instrucciones.
2. Después de crear el grupo de parámetros de clúster, abra la consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb. En el panel de navegación, elija Clusters (Clústeres) para añadir el nuevo grupo de parámetros a un clúster.
3. Elija el clúster al que desea asociar el grupo de parámetros. Elija Actions (Acciones), y, a continuación, elija Modify (Modificar) para modificar el clúster.
4. En Cluster options (Opciones de clúster), elija el nuevo grupo de parámetros al que desea asociar el clúster.
5. Elija Continue (Continuar) para ver un resumen de las modificaciones.
6. Después de verificar los cambios, puede aplicarlos inmediatamente o durante el siguiente período de mantenimiento en Scheduling of modifications (Programación de modificaciones).
7. Elija Modify cluster (Modificar clúster) para actualizar el clúster con el nuevo grupo de parámetros.

Using the AWS CLI

Antes de comenzar, asegúrese de haber creado un clúster de Amazon DocumentDB y un grupo de parámetros de clúster. Puede crear un clúster de Amazon DocumentDB mediante la AWS CLI `create-db-cluster` operación. Puede crear un grupo de parámetros de clúster mediante la AWS CLI `create-db-cluster-parameter-group` operación.

Para agregar su nuevo grupo de parámetros de clúster a su clúster, utilice la AWS CLI `modify-db-cluster` operación con los siguientes parámetros.

- --db-cluster-identifier: nombre del clúster (por ejemplo, `sample-cluster`).
- --db-cluster-parameter-group-name: nombre del grupo de parámetros al que quiere asociar el clúster (por ejemplo, `sample-parameter-group`).

Example

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --db-cluster-parameter-group-name sample-parameter-group
```

La salida de esta operación será similar a lo que se indica a continuación (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 de grupos de parámetros de clúster de Amazon DocumentDB

Puede hacer una copia de un grupo de parámetros de clúster en Amazon DocumentDB mediante Consola de administración de AWS o AWS Command Line Interface (AWS CLI).

Using the Consola de administración de AWS

El siguiente procedimiento le guía a través de la creación de un nuevo grupo de parámetros de clúster mediante una copia de un grupo de parámetros de clúster existente.

Copiar de un grupo de parámetros de clúster

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Grupos de parámetros.
3. En el panel Cluster parameter groups (Grupos de parámetros del clúster), seleccione el nombre del grupo de parámetros de clúster que quiera copiar.
4. Elija Actions (Acciones), y, a continuación, elija Copy (Copiar) para copiar ese grupo de parámetros.
5. En Copy options (Opciones de copia), introduzca un nombre y una descripción para el nuevo grupo de parámetros de clúster. A continuación, elija Copy (Copiar) para guardar los cambios.

Using the AWS CLI

Para realizar una copia de un grupo de parámetros de clúster, utilice la operación `copy-db-cluster-parameter-group` con los siguientes parámetros.

- **`--source-db-cluster-parameter-group-identifier`**: obligatorio. El nombre o nombre de recurso de Amazon (ARN) del grupo de parámetros de clúster del que desea realizar una copia.

  Si los grupos de parámetros del clúster de origen y de destino coinciden Región de AWS, el identificador puede ser un nombre o un ARN.

  Si los grupos de parámetros del clúster de origen y de destino son diferentes Regiones de AWS, el identificador debe ser un ARN.
- **`--target-db-cluster-parameter-group-identifier`**: obligatorio. El nombre o ARN del grupo de parámetros de clúster que se va a copiar.

  Restricciones:
  - No puede ser nulo ni estar vacío o en blanco.
  - Debe contener de 1 a 255 letras, números o guiones.

  - El primer carácter debe ser una letra.
  - No puede terminar con un guion ni contener dos guiones consecutivos.
- **`--target-db-cluster-parameter-group-description`**: obligatorio. Una descripción del usuario de la copia del grupo de parámetros de clúster.

Example

El siguiente código realiza una copia de `sample-parameter-group` y le asigna el nombre `sample-parameter-group-copy`.

Para 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"
```

Para 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"
```

La salida de esta operación será similar a lo que se indica a continuación (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"
    }
}
```

Restablecimiento de grupos de parámetros de clúster de Amazon DocumentDB

Puede restablecer algunos o todos los valores de los parámetros de un grupo de parámetros de clúster de Amazon DocumentDB a sus valores predeterminados mediante el Consola de administración de AWS o el AWS Command Line Interface (AWS CLI) para restablecer el grupo de parámetros del clúster.

Using the Consola de administración de AWS

Siga estos pasos para restablecer algunos o todos los valores de parámetros de un grupo de parámetros de clúster a sus valores predeterminados.

Restablecimiento de los valores de parámetros de un grupo de parámetros de clúster

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación del lado izquierdo de la consola, elija Parameter groups (Grupos de parámetros).
3. En el panel Cluster parameter groups (Grupos de parámetros del clúster), seleccione el nombre del grupo de parámetros de clúster que quiera reiniciar.
4. Elija Actions (Acciones), y, a continuación, elija Reset (Reiniciar) para restablecer ese grupo de parámetros.
5. En la página de confirmación de restablecimiento del grupo de parámetros del clúster resultante, confirme que desea restablecer todos los parámetros del clúster de ese grupo de parámetros a sus valores predeterminados. A continuación, seleccione Reset (Restablecer) para restablecer el grupo de parámetros. También puede seleccionar Cancel (Cancelar) para descartar los cambios.

Using the AWS CLI

Para restablecer algunos o todos los valores de parámetros de un grupo de parámetros de clúster a sus valores predeterminados, use la operación `reset-db-cluster-parameter-group` con los siguientes parámetros.

- **`--db-cluster-parameter-group-name`**: obligatorio. El nombre del grupo de parámetros de clúster que se va a restablecer.

- **`--parameters`**: opcional. Una lista de `ParameterName` y `ApplyMethod` del grupo de parámetros de clúster que se va a restablecer a sus valores predeterminados. Los parámetros estáticos se deben establecer en `pending-reboot` para que se apliquen la próxima vez que se reinicie la instancia o en la siguiente solicitud `reboot-db-instance`. Debe llamar a `reboot-db-instance` para cada instancia del clúster a la que desee aplicar el parámetro estático actualizado.

  Este parámetro y `--reset-all-parameters` se excluyen mutuamente: puede utilizar uno pero no ambos.

- **`--reset-all-parameters`** o **`--no-reset-all-parameters`**: opcional. Especifica si se deben restablecer todos los parámetros (`--reset-all-parameters`) o solo algunos de los parámetros (`--no-reset-all-parameters`) a sus valores predeterminados. El parámetro `--reset-all-parameters` y `--parameters` se excluyen mutuamente: puede utilizar uno pero no ambos.

  Cuando restablece todo el grupo, los parámetros dinámicos se actualizan inmediatamente. Los parámetros estáticos se establecen en `pending-reboot` para que se apliquen la próxima vez que se reinicie la instancia o en la siguiente solicitud `reboot-db-instance`. Debe llamar a `reboot-db-instance` para cada instancia del clúster a la que desee aplicar el parámetro estático actualizado.

Example

Ejemplo 1: restablecimiento de todos los parámetros a sus valores predeterminados

El siguiente código restablece todos los parámetros del grupo de parámetros de clúster `sample-parameter-group` a sus valores predeterminados.

Para Linux, macOS o Unix:

```
aws docdb reset-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group \
       --reset-all-parameters
```

Para Windows:

```
aws docdb reset-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group ^
```

```
       --reset-all-parameters
```

Ejemplo 2: restablecimiento de los parámetros especificados a sus valores predeterminados

El siguiente código restablece el parámetro `tls` del grupo de parámetros de clúster `sample-parameter-group` a su valor predeterminado.

Para 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
```

Para 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
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
{
       "DBClusterParameterGroupName": "sample-parameter-group"
   }
```

Reinicio de una instancia del clúster

Antes de que el valor de un parámetro estático cambie, la instancia del clúster debe reiniciarse. Reinicie cada instancia del clúster a la que desee aplicar el parámetro estático actualizado.

Para Linux, macOS o Unix:

```
aws docdb reboot-db-instance \
       --db-instance-identifier sample-cluster-instance
```

Para Windows:

```
aws docdb reboot-db-instance ^
```

```
    --db-instance-identifier sample-cluster-instance
```

Borrado de grupos de parámetros de clúster de Amazon DocumentDB

Puede eliminar un grupo de parámetros de clúster de Amazon DocumentDB personalizado mediante Consola de administración de AWS o AWS Command Line Interface ()AWS CLI. No se puede eliminar el grupo de parámetros del clúster `default.docdb4.0`.

Using the Consola de administración de AWS

Eliminación de un grupo de parámetros de clúster

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Grupos de parámetros.

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En el panel Parameter groups (Grupos de parámetros), seleccione el botón de opción situado a la izquierda del grupo de parámetros de clúster que desea eliminar.
4. Elija Acciones y, a continuación, elija Eliminar.
5. En el panel de confirmación Delete (Eliminar) elija Delete (Eliminar) para eliminar el grupo de parámetros del clúster. Para mantener el grupo de parámetros del clúster, elija Cancel (Cancelar).

Using the AWS CLI

Para eliminar un grupo de parámetros de clúster, utilice la operación `delete-db-cluster-parameter-group` con el siguiente parámetro.

- **`--db-cluster-parameter-group-name`**: obligatorio. El nombre del grupo de parámetros de clúster que se va a eliminar. Debe ser un grupo de parámetros de clúster existente. No se puede eliminar el grupo de parámetros del clúster *`default.docdb4.0`*.

Example- Eliminación de un grupo de parámetros de clúster

En el siguiente ejemplo, se presentan los tres pasos para eliminar un grupo de parámetros de clúster:

1. Buscar el nombre del grupo de parámetros de clúster que desea eliminar.
2. Eliminar el grupo de parámetros de clúster especificado.
3. Verificar que el grupo de parámetros de clúster se ha eliminado.

1. Busque el nombre del grupo de parámetros de clúster que desea eliminar.

El siguiente código muestra los nombres de todos los grupos de parámetros de clúster.

Para Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameter-groups \
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Para Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

El resultado de la operación anterior es una lista de los nombres de grupos de parámetros de clúster similar a la siguiente (formato JSON).

```
[
       [
           "default.docdb4.0"
       ],
       [
           "sample-parameter-group"
       ],
       [
           "sample-parameter-group-copy"
       ]
   ]
```

2. Elimine un grupo de parámetros de clúster.

El siguiente código elimina el grupo de parámetros de clúster `sample-parameter-group-copy`.

Para Linux, macOS o Unix:

```
aws docdb delete-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group-copy
```

Para Windows:

```
aws docdb delete-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group-copy
```

Esta operación no muestra ningún resultado.

3. Verifique que el grupo de parámetros de clúster especificado se ha eliminado.

El siguiente código muestra los nombres de todos los grupos de parámetros de clúster restantes.

Para Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameter-groups \
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Para Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

El resultado de la operación anterior es una lista de los grupos de parámetros de clúster similar a la siguiente (formato JSON). El grupo de parámetros de clúster que acaba de eliminar no debe estar en la lista.

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
[
       [
           "default.docdb4.0"
       ],
       [
           "sample-parameter-group"
```

```
        ]
    ]
```

Referencia de parámetros de clúster de Amazon DocumentDB

Cuando se modifica un parámetro dinámico y se guarda el grupo de parámetros de clúster, el cambio se aplica inmediatamente, sea cual sea el estado de Apply Immediately (Aplicar inmediatamente). Cuando se cambia un parámetro estático y se guarda el grupo de parámetros de clúster, el cambio de parámetros tiene efecto después de reiniciar manualmente la instancia. Puede reiniciar una instancia mediante la consola de Amazon DocumentDB o llamar de forma explícita a `reboot-db-instance`.

En la tabla siguiente, se muestran todos los parámetros que se aplican a todas las instancias de un clúster de Amazon DocumentDB.

Parámetros a nivel de clúster de Amazon DocumentDB

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
`audit_logs`	disabled	activado, desactivado, ddl, dml_read, dml_write, todos, ninguno	Sí	Dinámico	Cadena	Define si los registros de CloudWatch auditoría de Amazon están habilitados. • **`enabled`**—los registros CloudWatch de auditoría están

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
						habilitados. • **`disabled`**—los registros CloudWatch de auditoría están deshabilitados. • **`ddl`**: la auditoría de eventos de DDL está habilitada. • **`dml_read`**: la auditoría de eventos de escritura de DML está habilitada. • **`dml_write`**: la

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
						auditoría de eventos de escritura de DML está habilitada. • **`all`**: la auditoría de todos los eventos de la base de datos está habilitada. • **`none`**: la auditoría está deshabilitada.

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
`change_stream_log_retention_duration`	10800	3600-604800	Sí	Dinámico	Entero	Define la duración (en segundos) que se conserva y se puede consumir el registro del flujo de cambios.

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
`default_collection_compression`	disabled	activado, desactivado (Amazon DocumentDB 5.0)/ zstd, lz4, ninguno (Amazon) DocumentDB8.0	Sí	Dinámico	Cadena	Define la configuración de compresión predeterminada para las nuevas colecciones de un clúster • **`enabled`**—La compresión lz4 está habilitada de forma predeterminada en Amazon DocumentDB 5.0. La compresión zstd está habilitada de forma predeterminada en

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
						Amazon DocumentDB 8.0. • **`disabled, none`**— la compresión está deshabilitada de forma predeterminada (deshabilitada en Amazon DocumentDB 5.0, ninguna en Amazon DocumentDB 8.0).

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
`profiler`	disabled	habilitado, deshabilitado	Sí	Dinámico	Cadena	Habilita la creación de perfiles para operaciones lentas. • **`enabled`**— las operaciones que tardan más de un valor límite definido por el cliente (por ejemplo, 100 ms) se registran en Amazon Logs. CloudWatch • **`disabled`**— las operaciones lentas no se

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
						registran en los registros. CloudWatch
`profiler_sampling_rate`	1.0	0,0-1,0	Sí	Dinámico	Flotante	Define la velocidad de muestreo para las operaciones registradas.

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
`profiler_threshold_ms`	100	50-2147483646	Sí	Dinámico	Entero	Define el umbral de `profiler`. • Todas las operaciones superiores a `profiler_threshold_ms` las registradas en los CloudWatch registros.

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
`planner_version`	3.0	1.0, 2.0, 3.0	Sí	Dinámico	Flotante	Define la versión del planificador de consultas que se utilizará para las consultas. • **`1.0`**— Compatible con Amazon DocumentDB 5.0. • **`2.0`**— Mejoras para buscar y actualizar operadores. Compatible con Amazon DocumentDB 5.0. • **`3.0`**— Mejoras en los

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
						operadores de búsqueda, actualización y agregación. Planificador de consultas predeterminado en Amazon DocumentDB 8.0.

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
`tls`	enabled	habilitado, deshabilitado, fips-140-3, tls1.2+, tls1.3+	Sí	Estático	Cadena	Define si son obligatorias las conexiones de seguridad de la capa de transporte (TLS). • **`enabled`**: se requieren conexiones TLS para conectarse. Para Amazon DocumentDB 5.0 y versiones anteriores, habilita TLS 1.0 a 1.3, pero usa TLS 1.2 o superior.

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
						Amazon DocumentDB 5.0 solo admite TLS 1.2 y versiones posteriores, a partir de la versión secundaria 5.0.1, Amazon DocumentDB 8.0 y posteriores. • **`disabled`**: las conexiones TLS no se pueden usar para conectarse. • **`fips-140-3`** : para

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
						conectarse, se requieren conexiones TLS con atributos de Federal Information Processing Standards Publications (FIPS, Publicaciones de los Estándares de procesamiento de la información federal). El clúster solo acepta conexiones

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
						seguras según la publicación 140-3 de las FIPS. Esto solo se admite a partir de los clústeres de Amazon DocumentDB 5.0 (versión 3.0.3727 del motor) en estas regiones: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1,

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
						us-gov-west-1. • **`tls1.2+`**: se requieren conexiones TLS que utilicen la versión 1.2 o superior de TLS para conectarse. Esto solo se admite a partir de Amazon DocumentDB 4.0 (versión de motor 2.0.10980) y Amazon DocumentDB 5.0 (versión de

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
						motor 3.0.11051). • **`tls1.3+`**: se requieren conexiones TLS que utilicen la versión 1.3 o superior de TLS para conectarse. Esto solo se admite a partir de Amazon DocumentDB 4.0 (versión de motor 2.0.10980) y Amazon DocumentDB 5.0 (versión de

Parámetro	Predeterminado	Valores válidos	Modificable	Tipo de aplicación	Tipo de datos	Description (Descripción)
						motor 3.0.11051).
`ttl_monitor`	enabled	habilitado, deshabilitado	Sí	Dinámico	Cadena	Define si la monitorización de tiempo de vida (TTL) está habilitada para el clúster. • **`enabled`**: la supervisión de TTL está habilitada. • **`disabled`**: la supervisión de TTL está deshabilitada.

Modificación de parámetros de clúster de Amazon DocumentDB

En Amazon DocumentDB, los grupos de parámetros de clúster constan de parámetros que se aplican a todas las instancias que cree en el clúster. En los grupos de parámetros de clúster personalizados, puede modificar un valor de parámetro en cualquier momento o restablecer todos

los valores de parámetro a sus valores predeterminados para los grupos de parámetros que cree. En esta sección se describe cómo ver los parámetros que conforman un grupo de parámetros de clúster de Amazon DocumentDB y sus valores, y cómo cambiar o actualizar estos valores.

Los parámetros pueden ser dinámicos o estáticos. Cuando se modifica un parámetro dinámico y se guarda el grupo de parámetros de clúster, el cambio se aplica inmediatamente, sea cual sea el estado de `Apply Immediately`. Cuando se cambia un parámetro estático y se guarda el grupo de parámetros de clúster, el cambio de parámetros solo tiene efecto después de reiniciar manualmente la instancia.

Visualización de los parámetros de un grupo de parámetros de clúster de Amazon DocumentDB

Puede ver los parámetros de un Consola de administración de AWS clúster AWS CLI de Amazon DocumentDB y sus valores mediante el comando o.

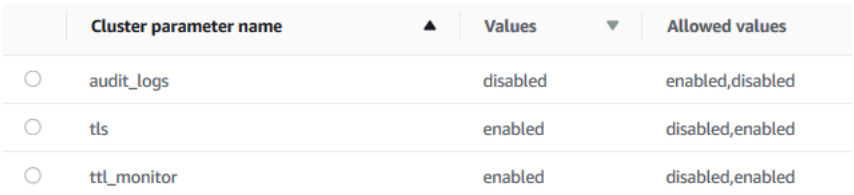
Using the Consola de administración de AWS

Visualización de los detalles de un grupo de parámetros de clúster

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Grupos de parámetros.

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En el panel Parameter groups (Grupos de parámetros), seleccione el nombre del grupo de parámetros de clúster cuyos detalles desee ver.
4. La página resultante muestra los siguientes valores para cada parámetro: el nombre del parámetro, el valor actual, los valores permitidos, si el parámetro es modificable, el tipo de aplicación, el tipo de datos y la descripción.



Using the AWS CLI

Para ver los parámetros de un grupo de parámetros de un clúster y sus valores, utilice la operación `describe-db-cluster-parameters` con los siguientes parámetros.

- **`--db-cluster-parameter-group-name`**: obligatorio. El nombre del grupo de parámetros de clúster para el que desea obtener una lista detallada de parámetros.
- **`--source`**: opcional. Si se especifica, devuelve solo los parámetros de un origen específico. Los orígenes de parámetros pueden ser `engine-default`, `system` o `user`.

Example

El siguiente código de ejemplo muestra los parámetros, junto con sus valores, del grupo de parámetros `custom3-6-param-grp`. Para obtener más información sobre el grupo de parámetros, omita la línea `--query`. Para obtener más información sobre todos los grupos de parámetros, omita la línea `--db-cluster-parameter-group-name`.

Para Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameters \
   --db-cluster-parameter-group-name custom3-6-param-grp \
   --query 'Parameters[*].[ParameterName,ParameterValue]'
```

Para Windows:

```
aws docdb describe-db-cluster-parameters ^
   --db-cluster-parameter-group-name custom3-6-param-grp ^
   --query 'Parameters[*].[ParameterName,ParameterValue]'
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
[
    [
        "audit_logs",
        "disabled"
    ],
    [
        "tls",
        "enabled"
    ],
    [
```

```
        "ttl_monitor",
        "enabled"
    ]
]
```

Modificación de los parámetros de un grupo de parámetros de clúster de Amazon DocumentDB

Puede modificar los parámetros de un grupo de parámetros mediante el comando Consola de administración de AWS o AWS CLI.

Using the Consola de administración de AWS

Actualización de los parámetros de un grupo de parámetros de clúster

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Grupos de parámetros.

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En el panel Parameter groups (Grupos de parámetros), elija el grupo de parámetros del clúster cuyos parámetros desea actualizar.
4. La página resultante muestra los parámetros y sus detalles correspondientes para este grupo de parámetros de clúster. Seleccione un parámetro para actualizar.
5. En la parte superior derecha de la página, elija Edit (Editar) para cambiar el valor del parámetro. Para obtener más información sobre los tipos de parámetros de clúster, consulte Referencia de parámetros de clúster de Amazon DocumentDB.
6. Realice el cambio y, a continuación, elija Modify cluster parameter (Modificar parámetro de clúster) para guardar los cambios. Para descartar los cambios, selecciona Cancel (Cancelar).

Using the AWS CLI

Para modificar los parámetros de un grupo de parámetros de clúster, utilice la operación `modify-db-cluster-parameter-group` con los siguientes parámetros.

- **`--db-cluster-parameter-group-name`**: obligatorio. El nombre del grupo de parámetros de clúster que va a modificar.
- **`--parameters`**: obligatorio. El parámetro o los parámetros que va a modificar. Cada entrada de parámetro debe incluir lo siguiente:
  - **`ParameterName`**: el nombre del grupo de parámetros de clúster que va a modificar.
  - **`ParameterValue`**: el valor nuevo de este parámetro de clúster.
  - **`ApplyMethod`**: cómo desea que se apliquen los cambios a este parámetro. Los valores permitidos son `immediate` y `pending-reboot`.

> **Note**
>
> Los parámetros con el `ApplyType` de `static` deben tener un `ApplyMethod` de `pending-reboot`.

Para cambiar los valores de los parámetros de un grupo de parámetros de clúster (AWS CLI)

En el siguiente ejemplo se cambia el nombre del parámetro `tls`.

1. Muestre los parámetros y sus valores de **`sample-parameter-group`**

   Para Linux, macOS o Unix:

   ```
   aws docdb describe-db-cluster-parameters \
       --db-cluster-parameter-group-name sample-parameter-group
   ```

   Para Windows:

   ```
   aws docdb describe-db-cluster-parameters ^
       --db-cluster-parameter-group-name sample-parameter-group
   ```

   La salida de esta operación será similar a lo que se indica a continuación (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. Modifique el parámetro **`tls`** para que su valor sea **`disabled`**. No se puede modificar `ApplyMethod` ya que `ApplyType` es `static`.

   Para 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"
```

   Para Windows:

```
aws docdb modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name sample-parameter-group ^
    --parameters "ParameterName=tls,ParameterValue=disabled,ApplyMethod=pending-
reboot"
```

   La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
{
```

```
    "DBClusterParameterGroupName": "sample-parameter-group"
}
```

3. Espere al menos 5 minutos.
4. Muestre los valores de parámetro de **`sample-parameter-group`**.

   Para Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

   Para Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

   La salida de esta operación será similar a lo que se indica a continuación (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"
        }
```

```
    ]
}
```

Descripción de los puntos de conexión de Amazon DocumentDB

Puede usar los puntos de conexión de Amazon DocumentDB (compatible con MongoDB) para conectarse a un clúster o a una instancia. Amazon DocumentDB tiene tres tipos diferentes de puntos de conexión, cada uno con su propia finalidad.

Temas

- Búsqueda de puntos de conexión de un clúster
- Búsqueda del punto de conexión de una instancia
- Conexión a puntos de conexión

Punto de conexión de clúster

Un punto de conexión de clúster es un punto de conexión de un clúster de Amazon DocumentDB que se conecta a la instancia principal actual del clúster. Cada clúster de Amazon DocumentDB tiene un solo punto de conexión de clúster y una sola instancia principal. En caso de que se produzca una conmutación por error, el punto de conexión del clúster se reasigna a la nueva instancia principal.

Punto de conexión del lector

Un punto de conexión del lector es un punto de conexión de un clúster de Amazon DocumentDB que se conecta a una de las réplicas disponibles de ese clúster. Cada clúster de base de datos de Amazon DocumentDB tiene un punto de conexión del lector. Si hay más de una réplica, el punto de conexión del lector dirige cada solicitud de conexión a una de las réplicas de Amazon DocumentDB.

Punto de conexión de instancia

Un punto de conexión de una instancia es aquel que se conecta a una instancia específica. Cada instancia de un clúster, independientemente de si es una instancia principal o de réplica, tiene su propio punto de conexión de instancia único. Es mejor no usar puntos de conexión de instancias en las aplicaciones. Esto se debe a que pueden cambiar los roles en caso de una conmutación por error, lo que obligaría a realizar modificaciones en el código de la aplicación.

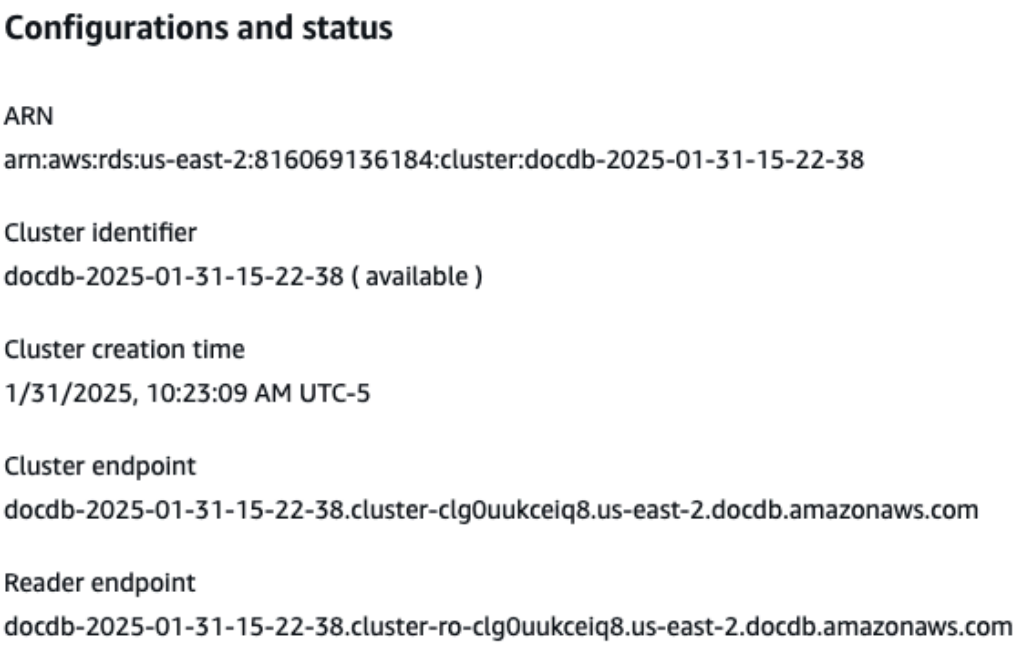
Búsqueda de puntos de conexión de un clúster

Puede buscar el punto de conexión de clúster de un clúster y el punto de conexión del lector mediante la consola de Amazon DocumentDB o la AWS CLI.

Using the Consola de administración de AWS

Cómo encontrar los puntos de conexión de un clúster mediante la consola:

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Clusters (Clústeres).
3. En la lista de clústeres, seleccione el nombre del clúster de su interés.
4. En la página de detalles del clúster, elija la pestaña Configuración. En la sección Configuraciones y estado, encontrará el Punto de conexión del clúster y el Punto de conexión del lector.



5. Para conectarse a este clúster, seleccione la pestaña Conectividad y seguridad. Localice la cadena de conexión del intérprete de comandos de mongo y una cadena de conexión que pueda utilizar en el código de la aplicación para conectarse a su clúster.



Using the AWS CLI

Para encontrar los puntos finales del clúster y del lector de su clúster mediante el AWS CLI, ejecute el `describe-db-clusters` comando con estos parámetros.

Parameters

- **`--db-cluster-identifier`**: opcional. Especifica el clúster para el que se van a devolver puntos de conexión. Si se omite, devuelve los puntos de conexión de hasta 100 clústeres.
- **`--query`**: opcional. Especifica los campos para mostrar. Sirve para reducir la cantidad de datos que se necesitan consultar para encontrar los puntos de conexión. Si se omite, se devuelve toda la información de un clúster.
- **`--region`**: opcional. Utilice el parámetro `--region` para especificar la región a la que desea aplicar el comando. Si se omite, se utiliza la región predeterminada.

Example

En el siguiente ejemplo, se devuelve el punto de conexión `DBClusterIdentifier` (punto de conexión del clúster) y `ReaderEndpoint` para `sample-cluster`.

Para Linux, macOS o Unix:

```
aws docdb describe-db-clusters \
   --region us-east-1 \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'
```

Para Windows:

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

```
    --region us-east-1 ^
    --db-cluster-identifier sample-cluster ^
    --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'
```

La salida de esta operación será similar a lo que se indica a continuación (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"
  ]
]
```

Ahora que tiene el punto de conexión del clúster, puede conectarse al clúster mediante `mongo` o `mongodb`. Para obtener más información, consulte Conexión a puntos de conexión.

Búsqueda del punto de conexión de una instancia

Puede encontrar los puntos de conexión para una instancia de base de datos mediante la consola de Amazon DocumentDB o la AWS CLI.

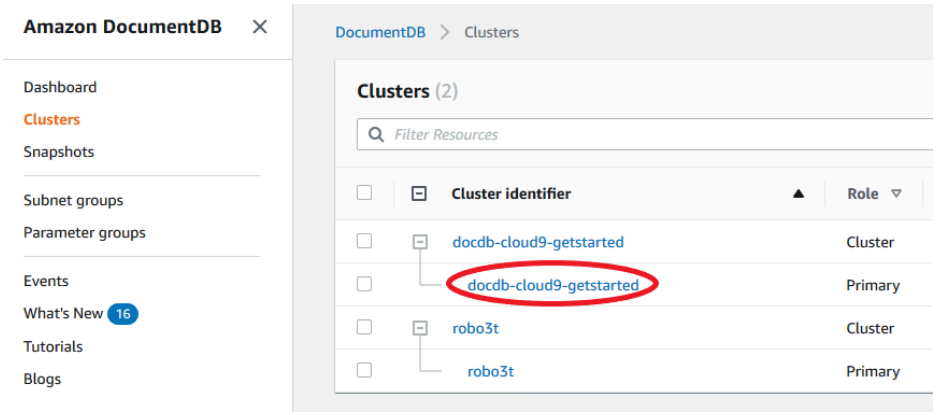
Using the Consola de administración de AWS

Búsqueda del punto de conexión de una instancia mediante la consola

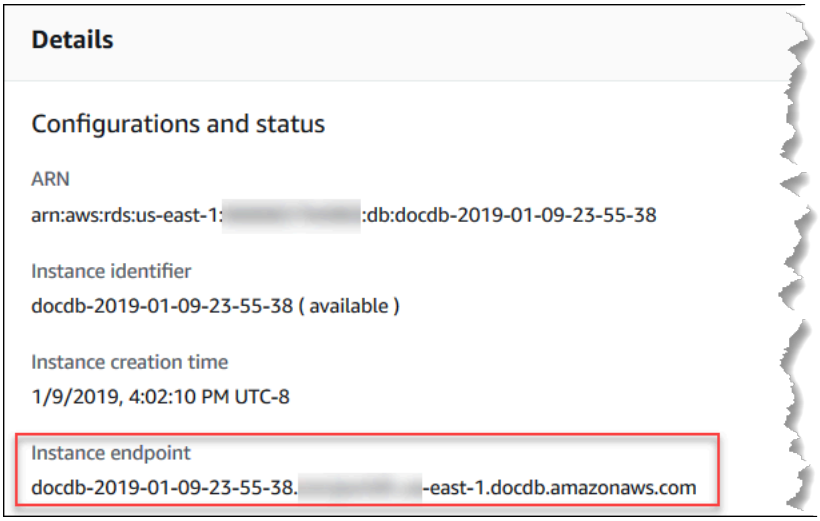
1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Clusters (Clústeres).

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En el cuadro de navegación de clústeres, verá la columna Identificador del clúster. Sus instancias se muestran en clústeres, de forma similar a la siguiente captura de pantalla.



4. Marque la casilla de verificación situada a la izquierda de la instancia de su interés.
5. Desplácese hacia abajo hasta la sección Details (Detalles) y localice el punto de conexión de la instancia.



6. Para conectarse a esta instancia, desplácese hacia abajo hasta la sección Connect (Conexión). Localice la cadena de conexión del intérprete de comandos de `mongo` y una cadena de conexión que pueda utilizar en el código de su aplicación para conectarse a su instancia.



Using the AWS CLI

Para encontrar el punto final de la instancia mediante el AWS CLI, ejecuta el siguiente comando con estos argumentos.

Argumentos

- **`--db-instance-identifier`**: opcional. Especifica la instancia para la que se va a devolver el punto de conexión. Si se omite, se devuelve el punto de conexión de hasta 100 instancias.
- **`--query`**: opcional. Especifica los campos para mostrar. Sirve para reducir la cantidad de datos que se necesitan consultar para encontrar los puntos de conexión. Si se omite, se devuelve toda la información de la instancia. El campo `Endpoint` tiene tres miembros, por lo que la consulta del siguiente ejemplo devuelve los tres miembros. Si solo está interesado en algunos de los miembros de `Endpoint` sustituya `Endpoint` en la consulta por los miembros de su interés, tal y como se muestra en el segundo ejemplo.
- **`--region`**: opcional. Utilice el parámetro `--region` para especificar la región a la que desea aplicar el comando. Si se omite, se utiliza la región predeterminada.

Example

Para Linux, macOS o Unix:

```
aws docdb describe-db-instances \
    --region us-east-1 \
    --db-instance-identifier sample-cluster-instance \
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint]'
```

Para Windows:

```
aws docdb describe-db-instances ^
    --region us-east-1 ^
    --db-instance-identifier sample-cluster-instance ^
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint]'
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
[
    [
```

```
        "sample-cluster-instance",
        {
            "Port": 27017,
            "Address": "sample-cluster-instance.corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
            "HostedZoneId": "Z2R2ITUGPM61AM"
        }
    ]
]
```

Reduciendo el resultado para eliminar el `HostedZoneId` del punto de conexión, puede modificar su consulta especificando `Endpoint.Port` y `Endpoint.Address`.

Para 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]'
```

Para Windows:

```
aws docdb describe-db-instances ^
    --region us-east-1 ^
    --db-instance-identifier sample-cluster-instance ^
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint.Port,Endpoint.Address]'
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
[
    [
        "sample-cluster-instance",
        27017,
        "sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com"
    ]
]
```

Ahora que tiene el punto de conexión de la instancia, puede conectarse a la instancia mediante `mongo` o `mongodb`. Para obtener más información, consulte Conexión a puntos de conexión.

Conexión a puntos de conexión

Cuando tenga su punto de conexión, ya sea del clúster o de la instancia, puede conectarse a él utilizando el intérprete de comandos de mongo o una cadena de conexión.

Conexión mediante el intérprete de comandos de mongo

Utilice la siguiente estructura para crear la cadena que necesita para conectarse a su clúster o instancia mediante el intérprete de comandos de mongo:

```
mongo \
    --ssl \
    --host Endpoint:Port \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

ejemplos de **mongo** shell

Conectarse a un clúster:

```
mongo \
    --ssl \
    --host sample-cluster.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

Conectarse a una instancia:

```
mongo \
    --ssl \
    --host sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

Conexión mediante una cadena de conexión

Utilice la siguiente estructura para crear la cadena de conexión que necesita para conectarse a su clúster o instancia.

```
mongodb://UserName:Password@endpoint:port?replicaSet=rs0&ssl_ca_certs=global-
bundle.pem
```

Ejemplos de cadenas de conexión

Conectarse a un clúster:

```
mongodb://UserName:Password@sample-cluster.cluster-corlsfccjozr.us-
east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=global-bundle.pem
```

Conectarse a una instancia:

```
mongodb://UserName:Password@sample-cluster-instance.cluster-corlsfccjozr.us-
east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=global-bundle.pem
```

Descripción de nombres de recurso de Amazon (ARN) en Amazon DocumentDB

Cada uno de los recursos que cree AWS se identifica de forma única con un nombre de recurso de Amazon (ARN). Para determinadas operaciones de Amazon DocumentDB (compatible con MongoDB), debe identificar de forma inequívoca un recurso de Amazon DocumentDB mediante su ARN. Por ejemplo, cuando añade una etiqueta a un recurso, debe proporcionar el ARN del recurso.

Temas

- Creación de un ARN para un recurso de Amazon DocumentDB
- Búsqueda de un ARN de recurso de Amazon DocumentDB

Creación de un ARN para un recurso de Amazon DocumentDB

Puede crear un ARN para un recurso de Amazon DocumentDB utilizando la siguiente sintaxis. Amazon DocumentDB comparte el formato de los ARN de Amazon Relational Database Service (Amazon RDS). Los ARN de Amazon DocumentDB contienen `rds` y no `docdb`

`arn:aws:rds:region:account_number:resource_type:resource_id`

Nombre de la región	Region	Zonas de disponibilidad (cálculo)
Este de EE. UU. (Ohio)	us-east-2	3
Este de EE. UU. (Norte de Virginia)	us-east-1	6
Oeste de EE. UU. (Oregón)	us-west-2	4
África (Ciudad del Cabo)	af-south-1	3
América del Sur (São Paulo)	sa-east-1	3
Asia-Pacífico (Hong Kong)	ap-east-1	3
Asia-Pacífico (Hyderabad)	ap-south-2	3
Asia-Pacífico (Malasia)	ap-southeast-5	3
Asia-Pacífico (Mumbai)	ap-south-1	3
Asia-Pacífico (Osaka)	ap-northeast-3	3
Asia-Pacífico (Seúl)	ap-northeast-2	4
Asia-Pacífico (Singapur)	ap-southeast-1	3
Asia-Pacífico (Sídney)	ap-southeast-2	3
Asia-Pacífico (Yakarta)	ap-southeast-3	3

Nombre de la región	Region	Zonas de disponibilidad (cálculo)
Asia-Pacífico (Melbourne)	ap-southeast-4	3
Asia-Pacífico (Tailandia)	ap-southeast-7	3
Asia-Pacífico (Tokio)	ap-northeast-1	3
Canadá (centro)	ca-central-1	3
Oeste de Canadá (Calgary)	ca-west-1	3
Región China (Pekín)	cn-north-1	3
China (Ningxia)	cn-northwest-1	3
Europa (Fráncfort)	eu-central-1	3
Europa (Zúrich)	eu-central-2	3
Europa (Irlanda)	eu-west-1	3
Europa (Londres)	eu-west-2	3
Europa (Milán)	eu-south-1	3
Europa (París)	eu-west-3	3
Europa (España)	eu-south-2	3
Europa (Estocolmo)	eu-north-1	3
México (centro)	mx-central-1	3
Medio Oriente (EAU)	me-central-1	3
Israel (Tel Aviv)	il-central-1	3

Nombre de la región	Region	Zonas de disponibilidad (cálculo)
AWS GovCloud (US-West)	`us-gov-west-1`	3
AWS GovCloud (US-East)	`us-gov-east-1`	3

> **Note**
>
> La arquitectura de Amazon DocumentDB separa el almacenamiento y la computación. Para la capa de almacenamiento, Amazon DocumentDB replica seis copias de los datos en tres zonas de AWS disponibilidad (AZ). Las AZ que se indican en la tabla anterior son la cantidad de AZ que puede utilizar en una región determinada para aprovisionar instancias de cómputo. Por ejemplo, si lanza un clúster de Amazon DocumentDB en ap-northeast-1, el almacenamiento se replicará de seis formas en tres zonas de disponibilidad, pero sus instancias de computación solo estarán disponibles en dos zonas de disponibilidad.

En la siguiente tabla se muestra el formato que debe utilizar al crear un ARN para un tipo de recurso concreto de Amazon DocumentDB. Amazon DocumentDB comparte el formato de los ARN de Amazon RDS. Los ARN de Amazon DocumentDB contienen `rds` y no `docdb`

Tipo de recurso	Formato de ARN / Ejemplo
Instancia (db)	`arn:aws:rds:` *`region`*`:`*`account_number`* `:db:`*`resource_id`* `arn:aws:rds:us-east-1:` *`1234567890`* `:db:`*`sample-db-instance`*
Clúster (`cluster`)	`arn:aws:rds:` *`region`*`:`*`account_number`* `:cluster:` *`resource_id`* `arn:aws:rds:us-east-1:` *`1234567890`* `:cluster:` *`sample-db-cluster`*

Tipo de recurso	Formato de ARN / Ejemplo
Grupo de parámetros del clúster (`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*
Grupo de seguridad (`secgrp`)	`arn:aws:rds:` *region*`:`*account_number* `:secgrp:`*resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:secgrp:`*sample-public-secgrp*
Instantánea de clúster (`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*
Grupo de subredes (`subgrp`)	`arn:aws:rds:` *region*`:`*account_number* `:subgrp:`*resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:subgrp:`*sample-subnet-10*

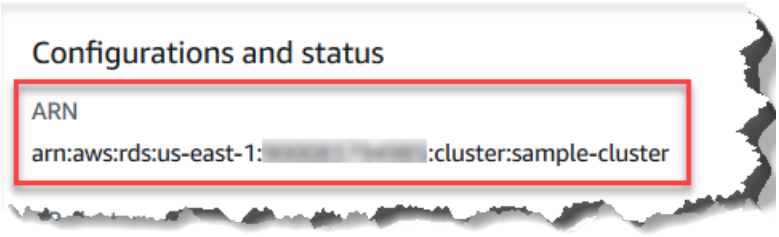
Búsqueda de un ARN de recurso de Amazon DocumentDB

Puede encontrar el ARN de un recurso de Amazon DocumentDB mediante Consola de administración de AWS el o el. AWS CLI

Using the Consola de administración de AWS

Para buscar un ARN mediante la consola, vaya al recurso cuyo ARN desea obtener y consulte los detalles de ese recurso.

Por ejemplo, puede obtener el ARN de un clúster desde la pestaña Configuración de los detalles del clúster. El ARN se encuentra en la sección Configuraciones y estado, como se muestra en la siguiente captura de pantalla.



Using the AWS CLI

Para obtener un ARN mediante un recurso concreto de Amazon DocumentDB, utilice `describe` la operación AWS CLI para ese recurso. La siguiente tabla muestra cada AWS CLI operación y la propiedad ARN que se usa con la operación para obtener un ARN.

AWS CLI Comando	Propiedad 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- Búsqueda del ARN de un clúster

La siguiente AWS CLI operación busca el ARN del clúster. `sample-cluster`

Para Linux, macOS o Unix:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterArn'
```

Para Windows:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterArn'
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
[
    "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster"
]
```

Example- Búsqueda de los ARN de varios grupos de parámetros

Para Linux, macOS o Unix:

```
aws docdb describe-db-cluster-parameter-groups \
   --query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

Para Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
   --query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

La salida de esta operación será similar a lo que se indica a continuación (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"
]
```

Etiquetado de recursos de Amazon DocumentDB

Puede utilizar etiquetas de Amazon DocumentDB (compatible con MongoDB) para agregar metadatos a los recursos de Amazon DocumentDB. Estas etiquetas se pueden usar con las políticas AWS Identity and Access Management (de IAM) para administrar el acceso a los recursos de Amazon DocumentDB y controlar qué acciones se pueden aplicar a los recursos. También puede utilizar etiquetas para hacer un seguimiento de los costos, agrupando los gastos correspondientes en recursos con etiquetas similares.

Puede etiquetar los siguientes recursos de Amazon DocumentDB:

- Clústeres
- instancias
- Snapshots
- Instantáneas del clúster
- Grupos de parámetros
- Grupos de parámetros de clústeres
- Grupos de seguridad
- Grupos de subredes

Información general de las etiquetas de recursos de Amazon DocumentDB

Las etiquetas de Amazon DocumentDB son pares nombre-valor que el usuario define y asocia a un recurso de Amazon DocumentDB. El nombre es la clave. Si lo desea puede proporcionar un valor para la clave o no. También puede usar etiquetas para asignar información arbitraria a un recurso de Amazon DocumentDB. Puede usar claves de etiqueta, por ejemplo, para definir una categoría, y el valor de la etiqueta puede ser un elemento dentro de esa categoría. Por ejemplo, puede definir una clave de etiqueta `project` y un valor de etiqueta de `Salix` para indicar que el recurso de Amazon DocumentDB se asigna al proyecto Salix. Asimismo, puede utilizar etiquetas para designar recursos de Amazon DocumentDB para pruebas o para producción mediante una clave como `environment=test` o `environment=production`. Se recomienda utilizar un conjunto coherente de claves de etiqueta que facilite el seguimiento de los metadatos asociados a los recursos de Amazon DocumentDB.

Puede usar etiquetas para organizar su AWS factura de manera que refleje su propia estructura de costos. Para ello, regístrate para recibir tu Cuenta de AWS factura con los valores clave de

las etiquetas incluidos. A continuación, para ver los costos de los recursos combinados, organice la información de facturación de acuerdo con los recursos con los mismos valores de clave de etiquetas. Por ejemplo, puede etiquetar varios recursos con un nombre de aplicación específico y luego organizar su información de facturación para ver los costos totales de la aplicación en distintos servicios. Para obtener más información, consulte Uso de etiquetas de asignación de costos en la AWS Guía del usuario de Administración de facturación y costos.

Cada recurso de Amazon DocumentDB tiene un conjunto de etiquetas que contiene todas las etiquetas asignadas a ese recurso. Un conjunto de etiquetas puede contener hasta 10 etiquetas, y también puede estar vacío. Si agrega una etiqueta a un recurso de Amazon DocumentDB con la misma clave que una etiqueta existente en el recurso, el nuevo valor sobrescribirá al antiguo.

AWS no aplica ningún significado semántico a sus etiquetas; las etiquetas se interpretan estrictamente como cadenas de caracteres. Amazon DocumentDB puede definir etiquetas en una instancia u otros recursos de Amazon DocumentDB, según la configuración utilizada al crear el recurso. Por ejemplo, Amazon DocumentDB podría añadir una etiqueta que indique que una instancia es para producción o para la realización de pruebas.

Puede añadir una etiqueta a una instantánea, pero la factura no reflejará esta agrupación.

Puede utilizar el Consola de administración de AWS o el AWS CLI para añadir, enumerar y eliminar etiquetas en los recursos de Amazon DocumentDB. Al utilizar el AWS CLI, debe proporcionar el nombre del recurso de Amazon (ARN) del recurso con el que quiere trabajar. Para obtener más información acerca de los ARN de Amazon DocumentDB, consulte Descripción de nombres de recurso de Amazon (ARN) en Amazon DocumentDB.

Restricciones de las etiquetas

Se aplican las siguientes restricciones de Amazon DocumentDB a las etiquetas:

- Cantidad máxima de etiquetas por recurso: 10
- Longitud máxima de la clave: 128 caracteres Unicode
- Longitud máxima del valor: 256 caracteres Unicode
- Caracteres válidos para la clave y el valor: letras mayúsculas y minúsculas en el juego de UTF-8 caracteres, dígitos, espacios y los siguientes caracteres: `_ . : / = + -` y `@` (expresión regular de Java:) `"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"`
- Las claves y los valores de las etiquetas distinguen entre mayúsculas y minúsculas.

- El prefijo `aws:` no se puede usar para las claves o valores de una etiqueta; está reservado para AWS.

Agregación y actualización de etiquetas en un recurso de Amazon DocumentDB

Puede añadir hasta 10 etiquetas a un recurso utilizando las Consola de administración de AWS o las AWS CLI.

Using the Consola de administración de AWS

El proceso para añadir una etiqueta a un recurso es similar independientemente de a qué recurso añada la etiqueta. En este ejemplo, se añade una etiqueta a un clúster.

Para añadir o actualizar etiquetas a un clúster utilizando la consola

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione clusters (clústeres).
3. Seleccione el nombre del clúster al que desea añadir etiquetas.
4. Desplácese hacia abajo hasta la sección Tags (Etiquetas) y, a continuación, elija Edit (Editar).
5. Para cada etiqueta que desee añadir a este recurso, haga lo siguiente:

    a. Para añadir una nueva etiqueta, escriba el nombre de la etiqueta en el cuadro Key (Clave). Para cambiar el valor de una etiqueta, busque el nombre de la etiqueta en la columna Key (Clave).

    b. Para asignar a la etiqueta un valor nuevo o actualizado, escriba un valor para la etiqueta en el cuadro Value (Valor).

    c. Si desea añadir más etiquetas, elija Add (Añadir). En caso contrario, cuando haya terminado, elija Save (Guardar).

Using the AWS CLI

El proceso para añadir una etiqueta a un recurso es similar independientemente de a qué recurso añada etiquetas. En este ejemplo, se añaden tres etiquetas a un clúster. La segunda etiqueta, `key2`, no tiene ningún valor.

Utilice la AWS CLI operación `add-tags-to-resource` con estos parámetros.

Parameters

- **`--resource-name`**: el ARN del recurso Amazon DocumentDB al que desea agregar etiquetas.
- **`--tags`**: una lista de etiquetas (par clave-valor) que desea agregar a este recurso en el formato. `Key=`*`key-name`*`,Value=`*`tag-value`*

Example

Para 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
```

Para 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
```

La operación `add-tags-to-resource` no produce ningún resultado. Para ver los resultados de la operación, utilice la operación `list-tags-for-resource`.

Lista de etiquetas en un recurso de Amazon DocumentDB

Puede utilizar Consola de administración de AWS o AWS CLI para obtener una lista de las etiquetas de un recurso de Amazon DocumentDB.

Using the Consola de administración de AWS

El proceso para mostrar las etiquetas de un recurso es similar independientemente de a qué recurso añada la etiqueta. En este ejemplo, se muestran las etiquetas de un clúster.

Para mostrar las etiquetas en un clúster mediante la consola

1. Abra la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione clusters (clústeres).

3. Seleccione el nombre del clúster para el que desea mostrar las etiquetas.
4. Para ver una lista de las etiquetas de este recurso, desplácese hacia abajo hasta la sección Tags (Etiquetas).

Using the AWS CLI

El proceso para mostrar las etiquetas de un recurso es similar independientemente del recurso cuyas etiquetas vaya a mostrar. En este ejemplo, se muestran las etiquetas de un clúster.

Utilice la AWS CLI operación `list-tags-for-resource` con estos parámetros.

Parameters

- **`--resource-name`**: obligatorio. El ARN del recurso Amazon DocumentDB del que desea ver las etiquetas.

Example

Para Linux, macOS o Unix:

```
aws docdb list-tags-for-resource \
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster
```

Para Windows:

```
aws docdb list-tags-for-resource ^
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
{
    "TagList": [
        {
            "Key": "key1",
            "Value": "value1"
        },
        {
            "Key": "key2",
            "Value": ""
```

```
        },
        {
            "Key": "key3",
            "Value": "value3"
        }
    ]
}
```

Eliminación de etiquetas de un recurso de Amazon DocumentDB

Puede utilizar Consola de administración de AWS o AWS CLI para eliminar etiquetas de los recursos de Amazon DocumentDB.

Using the Consola de administración de AWS

El proceso para eliminar etiquetas de un recurso es similar independientemente del recurso que vaya a eliminar. En este ejemplo, se eliminan las etiquetas de un clúster.

Para eliminar las etiquetas de un clúster mediante la consola

1. Abra la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione clusters (clústeres).
3. Seleccione el nombre del clúster cuyas etiquetas desea eliminar.
4. Desplácese hacia abajo hasta la sección Tags (Etiquetas) y, a continuación, elija Edit (Editar).
5. Si desea eliminar todas las etiquetas de este recurso, seleccione Remove all (Eliminar todas). De lo contrario, para cada etiqueta que desee eliminar de este recurso, haga lo siguiente:
    a. Localice el nombre de la etiqueta en la columna Key (Clave).
    b. Elija Remove (Eliminar) en la misma fila que la clave de etiqueta.
    c. Cuando termine, elija Guardar.

Using the AWS CLI

El proceso para eliminar una etiqueta de un recurso es similar independientemente del recurso del que vaya a eliminar una etiqueta. En este ejemplo, se elimina una etiqueta de un clúster.

Utilice la AWS CLI operación `remove-tags-from-resource` con estos parámetros.

- **`--resource-name`**: obligatorio. El ARN del recurso Amazon DocumentDB del que desea eliminar las etiquetas.
- **`--tag-keys`**: obligatorio. Una lista de las claves de etiqueta que desea quitar de este recurso.

Example

Para 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
```

Para Windows:

```
aws docdb remove-tags-from-resource ^
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tag-keys key1 key3
```

La operación `removed-tags-from-resource` no produce ningún resultado. Para ver los resultados de la operación, utilice la operación `list-tags-for-resource`.

Mantenimiento de Amazon DocumentDB

Amazon DocumentDB realiza periódicamente dos tipos de mantenimiento:

- El mantenimiento del clúster actualiza el motor de la base de datos. Las actualizaciones del motor incluyen correcciones de seguridad, correcciones de errores, nuevas funciones y otras mejoras del motor.
- El mantenimiento de la instancia actualiza el sistema operativo (SO) de la instancia.

Los parches del motor y las actualizaciones del sistema operativo utilizan las mismas tres categorías de ciclo de vida (opcionales , obligatorias y forzadas) con el mismo comportamiento de notificación y aplicación para cada categoría. Las versiones del motor también tienen una cuarta categoría: las versiones secundarias, a las que se actualizan manualmente. Las categorías son:

- Opcional: contiene mejoras no críticas. Sin fecha de solicitud automática ni notificación de AHD; solicítela cuando le convenga. (En el caso de las actualizaciones del sistema operativo, puedes suscribirte `RDS-EVENT-0230` para recibir una notificación cuando haya alguna disponible).
- Obligatorio: contiene correcciones de seguridad y otras soluciones críticas. Recibirá una notificación a través del Panel de estado (AHD) y por correo electrónico. Una acción obligatoria se aplica automáticamente durante el período de mantenimiento del clúster o instancia, una vez finalizado el período `AutoAppliedAfterDate` de mantenimiento. Puedes aplazarla cambiando la ventana de mantenimiento antes de esa fecha.
- Forzado: una solución poco frecuente y muy crítica. Auto-applies fuera de su período de mantenimiento una vez finalizado`ForcedApplyDate`. Amazon DocumentDB solo designa una acción forzada cuando no hay otra opción disponible.
- Versión secundaria (solo versiones del motor): una versión del motor numerada encima de la versión principal (por ejemplo,`5.0.1`). User-driven: se actualiza modificando la versión del motor del clúster. Nunca se aplica automáticamente; no hay notificación AHD. Las versiones secundarias no se publican para las versiones principales anteriores a la 5.0.

Los parches del motor se publican en una sola categoría (opcional, obligatoria o obligatoria) y permanecen allí. Las actualizaciones del sistema operativo avanzan: la mayoría comienzan como opcionales y, si no se aplican, pasan a ser obligatorias y, finalmente, forzadas. La fecha exacta depende del parche y se publica en la notificación de AHD y en los campos de fecha devueltos por `describe-pending-maintenance-actions` (consulteFechas de aplicación). Las notas de la versión de Amazon DocumentDB utilizan estos nombres de categoría al anunciar cambios en el motor.

Al aplicar cualquier parche al motor, el clúster se desconecta brevemente. El resto de este tema explica cómo funcionan las ventanas de mantenimiento, cómo encontrar el trabajo pendiente, cómo aplicar los parches del motor y las versiones secundarias, cómo funcionan las actualizaciones del sistema operativo y cómo tratar los clústeres globales.

Temas

- Acciones de mantenimiento para Amazon DocumentDB
- Numeración de versiones del motor
- Administración de los periodos de mantenimiento de Amazon DocumentDB
- Notificaciones de parches del motor de Amazon DocumentDB
- Visualización de las operaciones de mantenimiento pendientes de Amazon DocumentDB

- Actualizaciones del motor de Amazon DocumentDB
- Actualizaciones de la versión secundaria
- Actualizaciones del sistema operativo de Amazon DocumentDB
- User-initiated actualizaciones
- Aplicación de parches a clústeres globales

Acciones de mantenimiento para Amazon DocumentDB

Las siguientes acciones de mantenimiento se aplican a los clústeres de Amazon DocumentDB:

- `system-update`— Actualice el parche del motor para el clúster de Amazon DocumentDB. Para obtener más información, consulte Actualizaciones del motor de Amazon DocumentDB.
- `os-upgrade`— Actualice los sistemas operativos de todas las instancias de base de datos del clúster de Amazon DocumentDB mediante actualizaciones sucesivas. Para obtener más información, consulte Actualizaciones del sistema operativo de Amazon DocumentDB.

Las siguientes acciones de mantenimiento se aplican a las instancias de Amazon DocumentDB:

- `system-update`— Actualice el sistema operativo de la instancia de Amazon DocumentDB. En su lugar, le recomendamos que utilice la acción de mantenimiento a nivel de clústeros`-upgrade`. Para obtener más información, consulte Actualizaciones del sistema operativo de Amazon DocumentDB.

Numeración de versiones del motor

Amazon DocumentDB usa dos identificadores de versión independientes:

- Versión del motor: un número de tres partes en el formulario *`major`*`.`*`major`*`.`*`minor`* (por ejemplo, `5.0.0` o). `5.0.1` Las dos primeras partes (`5.0`) son la versión compatible con MongoDB; la tercera parte es la versión secundaria, que se incrementa cuando Amazon DocumentDB publica una versión secundaria que contiene correcciones de errores y mejoras importantes. Esta es la versión que especifica al crear o actualizar un clúster.
- Versión del parche del motor: un número independiente de tres partes en el formulario *`major`*`.0.`*`patch`* (por ejemplo`3.0.17983`) que identifica el nivel de parche aplicado al clúster.

El dígito del medio es siempre`0`. Las versiones de parches contienen correcciones críticas de seguridad y estabilidad.

Cuando la versión del parche del motor aparezca por sí sola, puede determinar la versión del motor a partir de su prefijo, tal y como se muestra en la tabla siguiente.

Prefijo de la versión del parche del motor	Versión del motor de Amazon DocumentDB
`1.0.`*x*	3.6
`2.0.`*x*	4.0
`3.0.`*x*	5.0
`4.0.`*x*	8.0

Para comprobar la versión del parche que está ejecutando su clúster, conéctese y ejecútelo`db.runCommand({getEngineVersion: 1})`.

> **Note**
>
> A partir de Amazon DocumentDB 5.0.2 y 8.0.2, `db.runCommand({getEngineVersion: 1})` devuelve la versión del motor y la versión del parche del motor como una sola cadena en el formulario. *`engine-version`*+*`engine-patch-version`* Por ejemplo, un clúster de Amazon DocumentDB 5.0.2 devuelve y un clúster de `5.0.2+3.0.12345` 8.0.2 devuelve. `8.0.2+4.0.12345` Los clústeres de versiones anteriores devuelven la versión del parche del motor por sí solos, por ejemplo. `3.0.12345`
> Si tiene scripts o monitores que analizan el resultado`getEngineVersion`, actualícelos para que utilicen el formato combinado.

Para ver la lista de las versiones de parches del motor publicadas y lo que contiene cada una de ellas, consulteNotas de la versión.

Administración de los periodos de mantenimiento de Amazon DocumentDB

Cada clúster y cada instancia tienen su propio período de mantenimiento semanal de 30 minutos, es decir, el período en el que se ejecutan las modificaciones programadas y los parches de software. La mayoría de los eventos se completan en 30 minutos; los más grandes pueden durar más tiempo.

Si no elige una ventana al crear el recurso, Amazon DocumentDB asigna una de forma aleatoria dentro de un bloque diario de 8 horas definido para la región, en un día seleccionado al azar. Elija ventanas que minimicen el impacto en su aplicación, por ejemplo, por la noche o los fines de semana.

Para las actualizaciones de los motores de bases de datos, Amazon DocumentDB usa la ventana del clúster, no las ventanas de las instancias individuales.

La siguiente tabla muestra los bloques de tiempo predeterminados por región.

Nombre de la región	Region	Bloque de tiempo en UTC
Este de EE. UU. (Ohio)	us-east-2	03:00-11:00
Este de EE. UU. (Norte de Virginia)	us-east-1	03:00-11:00
Oeste de EE. UU. (Oregón)	us-west-2	06:00-14:00
África (Ciudad del Cabo)	af-south-1	03:00-11:00
Asia Pacífico (Hong Kong)	ap-east-1	06:00-14:00
Asia-Pacífico (Hyderabad)	ap-south-2	06:30–14:30
Asia-Pacífico (Malasia)	ap-southeast-5	13:00-21:00
Asia-Pacífico (Mumbai)	ap-south-1	06:00-14:00
Asia-Pacífico (Osaka)	ap-northeast-3	12:00–20:00
Asia-Pacífico (Seúl)	ap-northeast-2	13:00-21:00
Asia-Pacífico (Singapur)	ap-southeast-1	14:00–22:00

Nombre de la región	Region	Bloque de tiempo en UTC
Asia-Pacífico (Sídney)	ap-southeast-2	12:00–20:00
Asia-Pacífico (Yakarta)	ap-southeast-3	De 08:00 a 16:00
Asia-Pacífico (Melbourne)	ap-southeast-4	11:00-19:00
Asia-Pacífico (Tailandia)	ap-southeast-7	15:00-23:00
Asia-Pacífico (Tokio)	ap-northeast-1	13:00-21:00
Canadá (centro)	ca-central-1	03:00-11:00
Oeste de Canadá (Calgary)	ca-west-1	18:00-02:00
China (Pekín)	cn-north-1	06:00-14:00
China (Ningxia)	cn-northwest-1	06:00-14:00
Europa (Fráncfort)	eu-central-1	21:00-05:00
Europa (Zúrich)	eu-central-2	02:00-10:00
Europa (Irlanda)	eu-west-1	22:00-06:00
Europa (Londres)	eu-west-2	22:00-06:00
Europa (Milán)	eu-south-1	02:00-10:00
Europa (París)	eu-west-3	23:59-07:29
Europa (España)	eu-south-2	02:00-10:00
Europa (Estocolmo)	eu-north-1	04:00 — 12:00
México (centro)	mx-central-1	03:00-11:00
Medio Oriente (EAU)	me-central-1	05:00-13:00
América del Sur (São Paulo)	sa-east-1	00:00-08:00

Nombre de la región	Region	Bloque de tiempo en UTC
Israel (Tel Aviv)	il-central-1	04:00-12:00
AWS GovCloud (US-East)	us-gov-east-1	17:00-01:00
AWS GovCloud (US-West)	us-gov-west-1	06:00-14:00

Cambio de los periodos de mantenimiento de Amazon DocumentDB

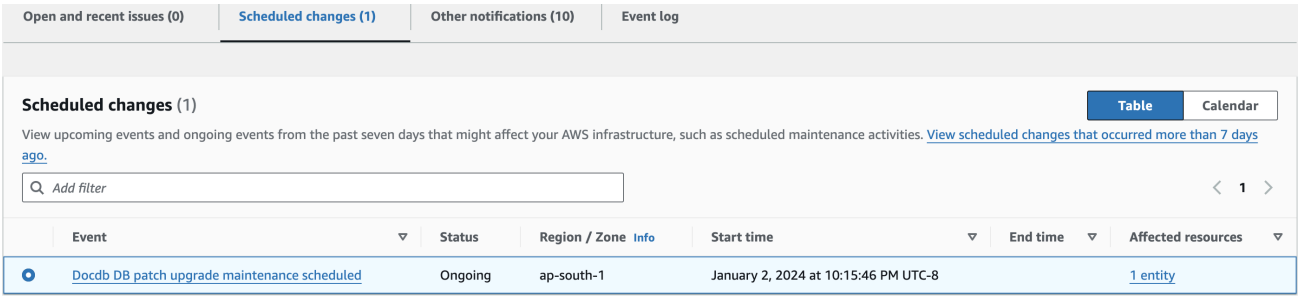
Elige la ventana con el tráfico más bajo que puedas y ajústala con el tiempo a medida que cambien tus patrones de tráfico. El clúster o la instancia no estarán disponibles durante ese período solo si un cambio en el sistema (una operación de almacenamiento a escala o un cambio de clase de instancia, por ejemplo) requiere una interrupción y solo durante el tiempo que ese cambio sea realmente necesario.

Para cambiar el periodo de mantenimiento

- Para un clúster, consulte Modificación de un clúster de Amazon DocumentDB.
- Para una instancia, consulte Modificación de una instancia de base de datos de Amazon DocumentDB.

Notificaciones de parches del motor de Amazon DocumentDB

Cuando el parche de motor necesario está disponible en una AWS región, todas las AWS cuentas que tengan un clúster de Amazon DocumentDB afectado en esa región reciben una notificación a través del Panel de estado (AHD) y por correo electrónico (que se envía a la dirección del usuario raíz de la AWS cuenta). Se envía una notificación por cada versión del motor de Amazon DocumentDB afectada. Puede encontrarlas en la sección Cambios programados del AHD. En cada notificación se indican los plazos de disponibilidad de los parches, el calendario de aplicación automática, los clústeres afectados y las notas de lanzamiento.



Los parches de motor necesarios tienen un único plazo de entrega de aproximadamente 30 días. Cuando un parche esté disponible en su región, Amazon DocumentDB envía la notificación descrita anteriormente. En ese momento, el parche `AutoAppliedAfterDate` estará listo aproximadamente 30 días después. Hasta esa fecha, el parche seguirá pendiente: puedes aplicarlo en cualquier momento o aplazarlo si cambias el período de mantenimiento del clúster a un día posterior. A partir de esa fecha`AutoAppliedAfterDate`, el parche se aplicará automáticamente durante la próxima ventana de mantenimiento del clúster.

Por ejemplo, un parche obligatorio que esté disponible el 1 de junio de 2026 caducará aproximadamente el 1 `AutoAppliedAfterDate` de julio de 2026. Recibirás la notificación el 1 de junio de 2026 y, si no tomas ninguna medida, el parche se aplicará automáticamente durante el primer período de mantenimiento del clúster, a partir del 1 de julio de 2026.

Tras recibir la notificación, tienes dos opciones: aplicar el parche por tu cuenta antes de la fecha de aplicación automática o esperar a que se aplique automáticamente durante el próximo período de mantenimiento (opción predeterminada). Para autoaplicarlo, abre la pestaña Mantenimiento y copias de seguridad del clúster y busca el tipo de entrada. `system-update`

> **Note**
>
> El estado de la notificación en el AHD permanece en curso hasta que Amazon DocumentDB publique otro parche del motor con una nueva versión del parche. Una vez aplicado el parche, la versión del parche del motor del clúster se actualiza para que coincida con la versión de la notificación. Verifique la nueva versión ejecutándoladb.`runCommand({getEngineVersion: 1})`.

Los parches opcionales y las nuevas versiones secundarias no generan notificaciones de AHD ni por correo electrónico. Para rastrearlos, consulte las notas de la versión de Amazon DocumentDB.

Los parches obligatorios (la categoría más infrecuente, reservada para las correcciones de seguridad más críticas) también se anuncian a través de AHD y por correo electrónico. A diferencia de los parches necesarios, se aplican fuera del período de mantenimiento, por lo que el ejemplo anterior sobre el tiempo de aplicación automática no se aplica.

Reaccionar a las notificaciones de parches mediante programación

AWS Health se integra con Amazon EventBridge, lo que le permite crear aplicaciones basadas en eventos para más de 20 destinos, incluido AWS Lambda Amazon Simple Queue Service (SQS). Para reaccionar de forma programática a la disponibilidad de los parches del motor, configúrala en función del evento. EventBridge `AWS_DOCDB_DB_PATCH_UPGRADE_MAINTENANCE_SCHEDULED` Desde allí, puede capturar los datos de los eventos, generar eventos adicionales, enviar notificaciones automáticas a través del AWS Console Mobile Application o realizar cualquier otra acción que necesite.

Si Amazon DocumentDB cancela un parche (poco frecuente), recibirá una notificación de AHD y un correo electrónico sobre la cancelación. Utilice el código del `AWS_DOCDB_DB_PATCH_UPGRADE_MAINTENANCE_CANCELLED` evento con Amazon EventBridge para gestionar este caso. Para obtener más información sobre la redacción de reglas, consulta la Guía del EventBridge usuario de Amazon.

Visualización de las operaciones de mantenimiento pendientes de Amazon DocumentDB

Usa el Consola de administración de AWS o el AWS CLI para comprobar qué mantenimiento está pendiente para un clúster o una instancia.

Las actualizaciones pendientes aparecen con el tipo de acción`system-update`, que abarca tanto los parches del motor como las actualizaciones del sistema operativo.

Cuando hay una actualización pendiente, puedes:

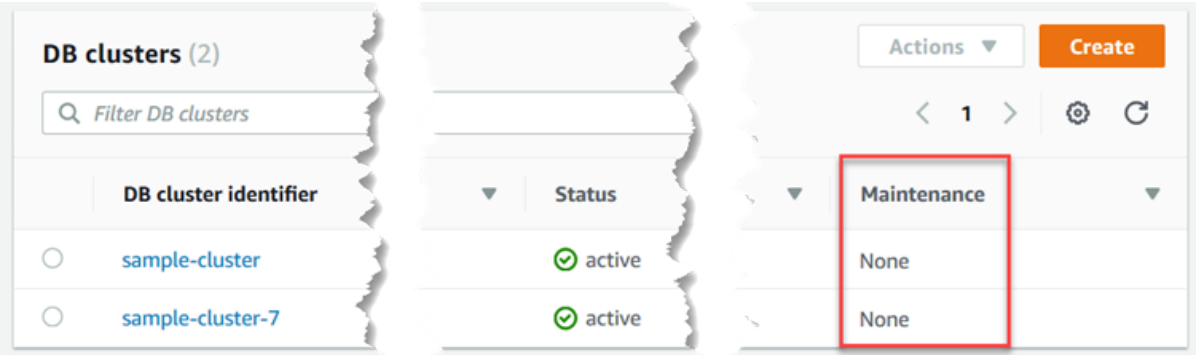
- Aplícala inmediatamente.
- Prográmela para la próxima ventana de mantenimiento.
- Aplazarlo (solo parches del motor y actualizaciones del sistema operativo) cambiando previamente el período de mantenimiento. `AutoAppliedAfterDate` Una vez que pase esa fecha, la acción se aplicará automáticamente durante el siguiente período de mantenimiento. Una vez que `ForcedApplyDate` pase, no es posible ningún otro aplazamiento.

> **Note**
>
> Si no realizas ninguna acción, las medidas de mantenimiento necesarias, como los parches necesarios para el motor, se aplicarán automáticamente durante el próximo período de mantenimiento. Los parches opcionales y las versiones secundarias nunca se aplican automáticamente.

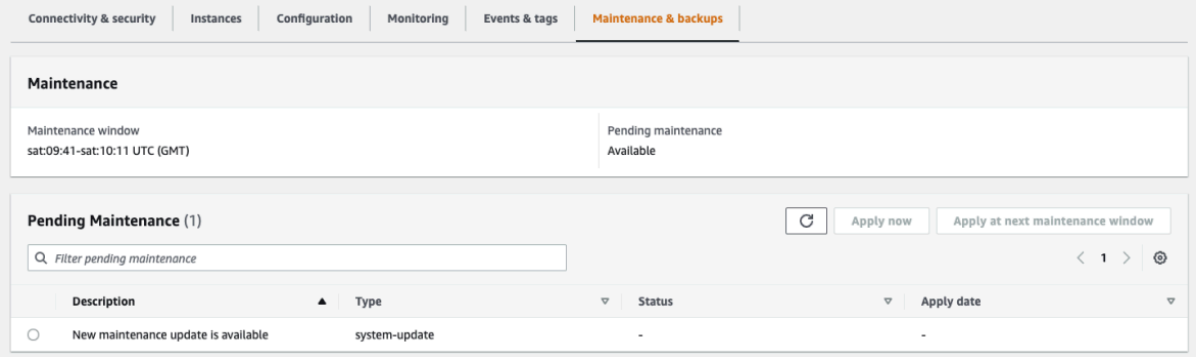
La ventana de mantenimiento controla cuándo se inician las operaciones pendientes, no cuánto tardan en completarse.

Using the Consola de administración de AWS

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Clusters (Clústeres).
3. La columna de mantenimiento del clúster muestra la ventana Disponible , Necesaria o Siguiente cuando hay una actualización pendiente.



4. Abra el clúster y, a continuación, seleccione Mantenimiento y copias de seguridad para ver los elementos pendientes de mantenimiento y tomar medidas al respecto.



Using the AWS CLI

Ejecuta `describe-pending-maintenance-actions` para ver lo que está pendiente. El siguiente ejemplo muestra una cuenta sin acciones pendientes.

```
aws docdb describe-pending-maintenance-actions
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
{
    "PendingMaintenanceActions": []
}
```

Una cuenta con una acción pendiente devuelve un resultado parecido al siguiente:

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

Puedes asignar el listado a clústeres específicos con `--filters` el formularioName=*`filter-name`*,Values=*`resource-id`*,.... El filtro aceptado `Name` es `db-cluster-id` el que contiene una lista de identificadores de clústeres o ARN.

Example

Para Linux, macOS o Unix:

```
aws docdb describe-pending-maintenance-actions \
```

```
    --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Para Windows:

```
aws docdb describe-pending-maintenance-actions ^
    --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Fechas de aplicación

Cada acción de mantenimiento pendiente tiene hasta tres fechas de aplicación. Aparecen en el AWS CLI resultado `describe-pending-maintenance-actions` e indican cuándo se ejecutará la acción. Los campos son `null` para mantenimiento opcional.

- **`CurrentApplyDate`**—cuando la acción esté programada para ejecutarse, ya sea ahora o en la próxima ventana de mantenimiento. Se rellena para las acciones obligatorias y forzadas.
- **`AutoAppliedAfterDate`**: la fecha a partir de la cual comienza la aplicación automática durante el período de mantenimiento del clúster o la instancia. Se rellena para indicar las acciones necesarias.
- **`ForcedApplyDate`**—la fecha límite fija. Después de esta fecha, la acción se ejecuta automáticamente, independientemente del período de mantenimiento. Se rellena para acciones forzadas.

Para aplazar una acción pendiente, traslade el período de mantenimiento a otro día anterior`AutoAppliedAfterDate`. Una vez `AutoAppliedAfterDate` aprobada, la acción se aplicará automáticamente durante el siguiente período de mantenimiento. Una vez `ForcedApplyDate` aprobada, no es posible ningún otro aplazamiento. El período exacto de aplazamiento varía según el parche; las fechas se publican en la notificación de AHD y en el AWS CLI resultado.

Actualizaciones del motor de Amazon DocumentDB

Cuando haya identificado un parche de motor pendiente, utilice uno de los siguientes procedimientos para aplicarlo o programarlo. Puede ejecutar estos procedimientos desde Consola de administración de AWS o desde AWS CLI.

Using the Consola de administración de AWS

Administración de la actualización de un clúster

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Clusters (Clústeres).
3. Seleccione el clúster que desea actualizar.
4. En el menú Acciones, elige una de las siguientes opciones:
   - Actualice ahora: ejecute el mantenimiento pendiente de inmediato.
   - Actualice en la siguiente ventana: ejecútela durante la próxima ventana de mantenimiento del clúster.

   También puede utilizar la opción Aplicar ahora o Aplicar en la próxima ventana de mantenimiento desde la sección Mantenimiento pendiente de la pestaña Mantenimiento y copias de seguridad del clúster (consulteVisualización de las operaciones de mantenimiento pendientes de Amazon DocumentDB).

   > **Note**
   >
   > Si no hay nada pendiente, todas estas opciones están inactivas.

Using the AWS CLI

Aplica una actualización pendiente con`apply-pending-maintenance-action`.

Parameters

- **`--resource-identifier`**—el nombre de recurso de Amazon (ARN) de Amazon DocumentDB del recurso al que se dirige la acción pendiente.
- **`--apply-action`**—La acción de mantenimiento pendiente de aplicar. Se usa `system-update` para aplicar un parche al motor.
- **`--opt-in-type`**: el tipo de solicitud de suscripción o si se debe deshacer una. Valores válidos:
  - `immediate`—solicítelo ahora. No se puede deshacer una vez enviado.

- `next-maintenance`—se aplica durante la próxima ventana de mantenimiento del recurso.
- `undo-opt-in`—cancelar una suscripción existente`next-maintenance`.

Example

Para 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
```

Para 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
```

Consulta la disponibilidad durante la aplicación de parches

Los motores 5.0 y 8.0 de Amazon DocumentDB conservan la disponibilidad de lectura durante la aplicación de parches cuando el clúster tiene varias instancias. Amazon DocumentDB parchea las instancias de los lectores de forma continua, dividiéndolas en tres grupos, de modo que los lectores restantes sigan distribuyendo el tráfico. El escritor no estará disponible durante un breve período mientras aplica los parches. Para lograr un tiempo de inactividad cero, defina sus preferencias de lectura de forma que la lectura recaiga en el escritor: `secondaryPreferred` o `primaryPreferred` trabajar; `primary` o `secondary` por sí sola, puede provocar un tiempo de inactividad.

Modo de preferencia de lectura	Durante la actualización del escritor	Durante la actualización del lector	Se necesita una cantidad mínima de lectores para que no haya ningún tiempo de inactividad en la lectura
`primary`	Read/write tiempo de inactividad	Sin impacto	N/A
`primaryPreferred`	Tiempo de inactividad de escritura	Sin impacto	1
`secondary`	Tiempo de inactividad de escritura	Tiempo de inactividad de lectura (si solo hay un lector)	2
`secondaryPreferred`	Tiempo de inactividad de escritura	Sin impacto	1
`nearest`	Tiempo de inactividad de escritura	Sin impacto	1

Mientras los lectores aplican los parches, el rendimiento general de lectura del clúster disminuye temporalmente. Para mantener un rendimiento estable, aprovisione lectores adicionales antes de la actualización y elimínelos una vez finalizada.

En los motores 3.6 y 4.0, estas funciones de disponibilidad de lectura no se aplican: un parche del motor provoca un tiempo de inactividad más prolongado, lo que afecta tanto a la lectura como a la escritura. Para actualizar a una versión principal que sí lo haga, consulte. Actualización local de la versión principal Amazon DocumentDB

Duración del tiempo de inactividad del parche

Engine-patch el tiempo de inactividad varía. Los factores más importantes son el uso de la CPU y la presión sobre la memoria de la instancia en el momento de la aplicación del parche, por lo que es importante ajustar el tamaño de las instancias. Para minimizar el tiempo de inactividad, ejecute

la versión más reciente del motor principal de Amazon DocumentDB y distribuya las instancias en varias zonas de disponibilidad.

Actualizaciones y reemplazos de parches

Amazon DocumentDB monitoriza los parches después de su lanzamiento. En el raro caso de que se identifique un problema, Amazon DocumentDB detiene la implementación mientras prepara una versión actualizada. Cuando esto ocurre, los clústeres que aún no han recibido el parche dejan de considerarlo una acción de mantenimiento disponible y se retira la correspondiente notificación de cambio programado que aparece en el. Panel de estado Los clústeres que ya ejecutan la versión afectada siguen funcionando con normalidad y no requieren ninguna acción por tu parte.

En breve recibirá un parche actualizado. Cuando esté disponible en su región, recibirá una nueva notificación por correo electrónico, tal Panel de estado y como se describe enNotificaciones de parches del motor de Amazon DocumentDB.

Actualizaciones de la versión secundaria

Amazon DocumentDB publica versiones secundarias además de la versión principal 5.0 y posteriores (por ejemplo,`5.0.1`). Las versiones secundarias no se publican para las versiones principales anteriores a la 5.0. Las versiones secundarias se comportan de manera diferente a los parches de motor obligatorios y opcionales:

- No aparecen como una acción de mantenimiento pendiente y nunca se aplican automáticamente.
- No generan notificaciones AHD ni por correo electrónico. Las nuevas versiones secundarias se anuncian en las notas de lanzamiento de Amazon DocumentDB.
- Para realizar la actualización, modifique la versión del motor del clúster (inmediatamente o durante el siguiente período de mantenimiento). Las actualizaciones de versiones secundarias requieren un breve tiempo de inactividad y son unidireccionales; no se puede cambiar a una versión secundaria anterior. En el caso de los clústeres globales, actualiza los clústeres secundarios antes que los principales.

Leer más:Actualización de la versión secundaria de Amazon DocumentDB.

Actualizaciones del sistema operativo de Amazon DocumentDB

Ocasionalmente, las instancias necesitan actualizaciones del sistema operativo. Amazon DocumentDB actualiza el sistema operativo para mejorar el rendimiento y reforzar la seguridad.

Las actualizaciones del sistema operativo no modifican la versión del motor del clúster ni la clase de instancia. Al igual que los parches del motor, las actualizaciones del sistema operativo utilizan el ciclo de vida opcional, obligatorio o forzado que se describe al principio de este tema; a diferencia de los parches del motor, una actualización del sistema operativo puede pasar de una categoría a otra con el tiempo si se aplaza. Aplica las actualizaciones del sistema operativo tan pronto como estén disponibles y configura los periodos de mantenimiento de los clústeres e instancias según las necesidades de tu empresa.

Usa la acción de `os-upgrade` mantenimiento a nivel de clúster para aplicar las actualizaciones del sistema operativo en todas las instancias de un clúster. Amazon DocumentDB actualiza las instancias de forma continua, unas cuantas a la vez, y actualiza la instancia principal en último lugar para minimizar las conmutaciones por error. La actualización se ejecuta durante el período de mantenimiento del clúster (no durante el período de mantenimiento de las instancias individuales) que usted configure.

Cuando una instancia recibe una actualización del sistema operativo, su caché de búfer comienza a vaciarse. Hasta que el conjunto de trabajo no se rellene desde el volumen de almacenamiento, las consultas de esa instancia pueden experimentar una latencia mayor y menor`BufferCacheHitRatio`.

Cuando Amazon DocumentDB actualiza la instancia principal, una conmutación por error hace que una réplica pase a ser la nueva instancia principal. Utilice el punto de enlace del clúster para que la aplicación gestione esta situación de forma transparente. Para mantener la disponibilidad de lectura mientras se actualizan las instancias, defina su preferencia de lectura para `secondaryPreferred` `primaryPreferred` que las lecturas se basen en una instancia disponible. Mantén los posibles objetivos de conmutación por error (réplicas con el nivel de prioridad más alto) en la misma clase de instancia que la instancia principal. Esto evita la degradación del rendimiento de escritura después de la promoción. Para obtener más información, consulte Conmutación por error de Amazon DocumentDB.

Tanto las acciones a nivel de clúster como a nivel `os-upgrade` de instancia pueden aparecer simultáneamente como `system-update` acciones disponibles. `describe-pending-maintenance-actions` Sin embargo, no puede programar ambas al mismo tiempo. Si `system-update` las acciones a nivel de instancia están programadas activamente en cualquier instancia, debes cancelarlas o completarlas antes de programar la acción a nivel de clúster, y viceversa`os-upgrade`.

> **Important**
>
> Su instancia de Amazon DocumentDB se desconecta para actualizar el sistema operativo. Multi-instance los clústeres minimizan el impacto. Si ejecutas un clúster de una sola instancia, puedes agregar temporalmente un clúster secundario para la actualización y eliminarlo después. El secundario incurre en los cargos habituales mientras exista.

> **Note**
>
> La `system-update` acción a nivel de instancia sigue disponible por motivos de compatibilidad con versiones anteriores. Si tiene que usarla, actualice primero las réplicas y al final la principal; evite aplicarles parches simultáneamente, ya que la conmutación por error durante el parche puede prolongar el tiempo de inactividad.

Para recibir un evento cuando llegue una nueva actualización opcional del sistema operativo, suscríbase a la categoría de eventos de aplicación de parches de `RDS-EVENT-0230` seguridad. Para obtener más información, consulte Suscripción a eventos de Amazon DocumentDB.

> **Note**
>
> Es posible que sea necesario mantenerse al día con las actualizaciones opcionales y obligatorias para garantizar el cumplimiento. Aplica `os-upgrade` acciones de forma rutinaria durante los períodos de mantenimiento.

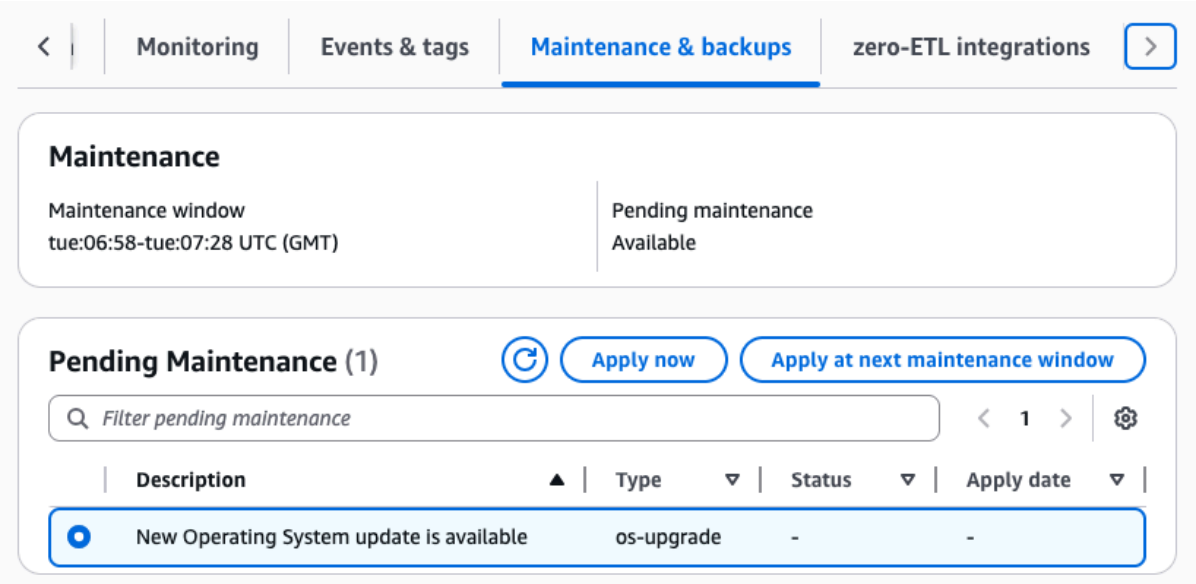
Las actualizaciones del sistema operativo están vinculadas a clases de instancias específicas, por lo que diferentes instancias son aptas en momentos distintos. Si tu clúster no incluye el parche de motor más reciente, es posible que la actualización del sistema operativo no aparezca. Aplica primero el último parche del motor (consultaActualizaciones del motor de Amazon DocumentDB).

Usa el Consola de administración de AWS o AWS CLI para comprobar si hay una actualización disponible.

Using the Consola de administración de AWS

Para comprobar si hay una actualización del sistema operativo desde la consola:

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, elija Clústeres y, a continuación, seleccione el nombre del clúster.
3. Selecciona la pestaña Mantenimiento y copias de seguridad.
4. En Mantenimiento pendiente, la `os-upgrade` acción aparece si hay una actualización del sistema operativo disponible.



5. Seleccione la `os-upgrade` acción y elija Aplicar ahora o Aplicar en la próxima ventana de mantenimiento. Si el valor es la siguiente ventana, puedes aplazar la actualización mientras la acción no se haya iniciado.

Using the AWS CLI

Comprueba si hay una actualización pendiente del sistema operativo:

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

Las actualizaciones del sistema operativo aparecen a nivel de clúster como `os-upgrade` y a nivel de instancia como`system-update`. Utilice la acción a nivel de clústeros`-upgrade`.

Example

En el siguiente ejemplo, se aplica la actualización del sistema operativo de forma inmediata.

Para 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
```

Para 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 actualizaciones

Algunos cambios los inicias tú mismo, por ejemplo, cambiar una clase de instancia por una con más o menos memoria o cambiar el grupo de parámetros del clúster. Amazon DocumentDB los trata de manera diferente a las actualizaciones que inicia. Para obtener más información, consulte:

- Modificación de un clúster de Amazon DocumentDB
- Modificación de una instancia de base de datos de Amazon DocumentDB

Para enumerar los cambios iniciados por el usuario que aún están pendientes:

Example

Para enumerar los cambios pendientes iniciados por el usuario para tus instancias

Para Linux, macOS o Unix:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

Para Windows:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

En este ejemplo, `sample-cluster-instance` tiene un cambio pendiente en`db.r5.xlarge`; no `sample-cluster-instance-2` tiene ninguno.

```
[
    [
        "sample-cluster",
        "sample-cluster-instance",
        {
            "DBInstanceClass": "db.r5.xlarge"
        }
    ],
    [
        "sample-cluster",
        "sample-cluster-instance-2",
        {}
    ]
]
```

Aplicación de parches a clústeres globales

En un clúster global, cada clúster miembro (principal y secundario) se actualiza durante su propio período de mantenimiento. Cuando el parche de motor necesario esté disponible en cada región, recibirá una notificación por correo electrónico y de AHD. Los parches opcionales y las nuevas versiones secundarias no generan notificaciones; consulte las notas de la versión de Amazon DocumentDB para obtener información al respecto.

Si se aplica por sí mismo, aplique siempre parches primero a los secundarios y al final al primario. Este orden mantiene la conmutación por error y la conmutación disponibles durante toda la implementación.

> **Important**
>
> Si primero parcheas la versión principal por error, actualiza todas las versiones secundarias a la misma versión lo antes posible. La conmutación por error y la conmutación permanecen deshabilitadas hasta que todos los clústeres estén en la misma versión.

Si no realiza ninguna acción, el parche se aplica automáticamente durante el siguiente período de mantenimiento de cada clúster: primero los secundarios y luego el principal en su ventana una vez que los secundarios hayan terminado.

Mantenga los clústeres de bases de datos principales y secundarios en la misma versión. La conmutación por error gestionada entre regiones solo funciona en una base de datos global cuando

todos los clústeres comparten la misma versión del motor y el mismo nivel de parche. Lo mismo ocurre si agregas una nueva versión del motor secundaria que usa una versión del motor más reciente que la principal: crea nuevas secundarias en la versión principal antes de unirlas a la base de datos global.

Tras recibir la notificación de un parche, actualice la versión principal y secundaria a la versión más reciente lo antes posible para que la conmutación por error y la conmutación sigan funcionando. Si se rechaza una solicitud de conmutación por error o conmutación, compare las versiones de los parches del motor en los distintos clústeres; si no coinciden, aplique el parche disponible en los clústeres que no coincidan.

Descripción de los roles vinculados a servicios

Amazon DocumentDB (compatible con MongoDB) utiliza funciones vinculadas a servicios AWS Identity and Access Management (IAM). Un rol vinculado a un servicio es un tipo único de rol de IAM que está vinculado directamente a Amazon DocumentDB. Service-linked Amazon DocumentDB predefine los roles e incluyen todos los permisos que el servicio necesita para llamar a otros AWS servicios en su nombre.

Un rol vinculado a un servicio simplifica el uso de Amazon DocumentDB porque ya no tendrá que agregar manualmente los permisos requeridos. Amazon DocumentDB define los permisos de sus roles vinculados al servicio y, a menos que esté definido de otra manera, solo Amazon DocumentDB puede asumir sus roles. Los permisos definidos incluyen las políticas de confianza y de permisos, y que la política de permisos no se pueda asociar a ninguna otra entidad de IAM.

Las funciones se pueden eliminar únicamente después de eliminar primero sus recursos relacionados. De esta forma, se protegen los recursos de Amazon DocumentDB, ya que se evita que se puedan eliminar accidentalmente permisos de acceso a los recursos.

Para obtener información acerca de otros servicios que son compatibles con roles vinculados a servicios, consulte Servicios de AWS que funcionan con IAM y busque los servicios en los que se indica Sí en la columna Service-Linked Rol vinculado a servicios. Elija una opción Sí con un enlace para ver la documentación acerca del rol vinculado al servicio en cuestión.

Permisos de roles vinculados a servicios para Amazon DocumentDB

Amazon DocumentDB (compatible con MongoDB) utiliza el rol vinculado a servicios denominado AWSServiceRoleForRDS para permitir que Amazon DocumentDB llame a los AWS servicios en nombre de sus clústeres.

El rol AWSServiceRoleForRDS vinculado a un servicio confía en que los siguientes servicios asuman el rol:

- `docdb.amazonaws.com`

La política de permisos del rol permite que Amazon DocumentDB realice las siguientes acciones en los recursos especificados:

- Acciones en `ec2`:
  - `AssignPrivateIpAddresses`
  - `AuthorizeSecurityGroupIngress`
  - `CreateNetworkInterface`
  - `CreateSecurityGroup`
  - `DeleteNetworkInterface`
  - `DeleteSecurityGroup`
  - `DescribeAvailabilityZones`
  - `DescribeInternetGateways`
  - `DescribeSecurityGroups`
  - `DescribeSubnets`
  - `DescribeVpcAttribute`
  - `DescribeVpcs`
  - `ModifyNetworkInterfaceAttribute`
  - `RevokeSecurityGroupIngress`
  - `UnassignPrivateIpAddresses`
- Acciones en `sns`:
  - `ListTopic`
  - `Publish`
- Acciones en `cloudwatch`:
  - `PutMetricData`
  - `GetMetricData`
  - `CreateLogStream`
  - `PullLogEvents`

- `DescribeLogStreams`
- `CreateLogGroup`

> **Note**
>
> Debe configurar permisos para permitir a una entidad de IAM (como un usuario, grupo o rol) crear, editar o eliminar un rol vinculado a servicios. Podría encontrarse con el siguiente mensaje de error:
> Unable to create the resource. Verify that you have permission to create service linked role. Otherwise wait and try again later.
> Si aparece este error, asegúrese de que tiene los siguientes permisos habilitados:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
> AWSServiceRoleForRDS",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"rds.amazonaws.com"
>         }
>     }
> }
> ```

Para obtener más información, consulte los permisos de Service-Linked rol en la Guía del usuario de IAM.

Creación de un rol vinculado a servicios para Amazon DocumentDB

No necesita crear manualmente un rol vinculado a servicios. Al crear un clúster, Amazon DocumentDB se encarga de crear de nuevo el rol vinculado a servicios en su nombre.

Si elimina este rol vinculado a un servicio y necesita crearlo de nuevo, puede utilizar el mismo proceso para volver a crear el rol en su cuenta. Al crear un clúster, Amazon DocumentDB se encarga de crear de nuevo el rol vinculado a servicios en su nombre de nuevo.

Modificación de un rol vinculado a servicios para Amazon DocumentDB

Amazon DocumentDB no permite modificar el rol vinculado al AWSServiceRoleForRDS servicio. Después de crear un rol vinculado a un servicio, no podrá cambiar el nombre del rol, ya que varias entidades podrían hacer referencia al mismo. Sin embargo, puede modificar la descripción del rol utilizando IAM. Para obtener más información, consulte Edición de un Service-Linked rol en la Guía del usuario de IAM.

Eliminación de un rol vinculado a servicios para Amazon DocumentDB

Si ya no necesita usar una función o un servicio que requiera un rol vinculado a un servicio, elimínelo. De esa forma, no tendrás una entidad sin usar que no se supervise o mantenga activamente. Sin embargo, debe eliminar todos los clústeres para poder eliminar el rol vinculado al servicio.

Limpieza de un rol vinculado a servicios para Amazon DocumentDB

Antes de poder utilizar IAM para eliminar un rol vinculado a un servicio, primero debe confirmar que dicho rol no tiene sesiones activas y eliminar los recursos que utiliza.

Para comprobar si el rol vinculado al servicio tiene una sesión activa mediante la consola

1. Inicia sesión en Consola de administración de AWS y abre la consola de IAM enhttps://console.aws.amazon.com/iam/.
2. En el panel de navegación de la consola de IAM, seleccione Funciones y, a continuación, elija el nombre (no la casilla de verificación) de la AWSServiceRoleForRDS función.
3. En la página Resumen del rol seleccionado, seleccione la pestaña Asesor de acceso.
4. En la pestaña Access Advisor, revise la actividad reciente del rol vinculado al servicio.

> **Note**
>
> Si no está seguro de si Amazon DocumentDB utiliza el AWSServiceRoleForRDS rol, puede intentar eliminarlo. Si el servicio está utilizando el rol, este no podrá eliminarse y podrá ver las regiones en las que se está utilizando. Si el rol se está utilizando, debe esperar que la sesión finalice para poder eliminarlo. No se puede revocar la sesión de un rol vinculado a servicios.

Si desea eliminar el AWSServiceRoleForRDS rol, primero debe eliminar todas las instancias y clústeres. Para obtener información acerca de cómo eliminar instancias y clústeres, consulte los siguientes temas:

- Eliminación de una instancia de Amazon DocumentDB
- Eliminación de un clúster de Amazon DocumentDB

Regiones compatibles con los roles vinculados al servicio de Amazon DocumentDB

Amazon DocumentDB admite el uso de roles vinculados a servicios en todas las regiones en las que se encuentra disponible el servicio. Para obtener más información, consulte https://docs.aws.amazon.com/documentdb/latest/devguide/regions-and-azs.html#regions-and-azs-availability.

Optimización del rendimiento

Este capítulo proporciona orientación sobre la optimización del rendimiento de Amazon DocumentDB y la solución de problemas de rendimiento comunes.

Temas

- Problemas comunes de rendimiento
- Anti-patterns
- Consejos para mejorar el rendimiento
- Análisis del plan de consultas
- Index-only escaneos (consultas cubiertas)
- Planificar filtros de caché
- Prácticas recomendadas para Amazon DocumentDB
- Administración de índices de Amazon DocumentDB
- Gestión de la compresión de documentos a nivel de colección
- Administración de la compresión basada en diccionarios en Amazon DocumentDB 8.0

Problemas comunes de rendimiento

En esta sección se describen los problemas de rendimiento más comunes que pueden surgir con Amazon DocumentDB y cómo resolverlos.

Temas

- Alta utilización de la CPU
- Problemas de conexión con Amazon DocumentDB
- La consulta se ejecuta lentamente

Alta utilización de la CPU

El uso elevado de la CPU es uno de los problemas que puede encontrar en algún momento al trabajar con Amazon DocumentDB. Esta sección proporciona información para resolver el problema.

Identificación: localice el problema

El uso elevado de la CPU en las instancias de Amazon DocumentDB provoca lentitud en las aplicaciones al aumentar las latencias en todas las operaciones de la base de datos. Cuando el uso de la CPU de Amazon DocumentDB supera los niveles óptimos, los tiempos de ejecución de las consultas aumentan, el establecimiento de la conexión tarda más y la capacidad de respuesta de las aplicaciones disminuye. Los picos de CPU en la instancia principal ralentizan las operaciones de escritura, mientras que los picos en las réplicas aumentan la latencia de las operaciones de lectura.

Algunas de las causas más comunes del uso elevado de la CPU en Amazon DocumentDB podrían ser:

- Escaneos de colecciones: las consultas sin los índices adecuados obligan a realizar escaneos de colecciones completos y consumen una cantidad excesiva de CPU para examinar todos los documentos.
- Picos de conexión: los picos repentinos de conexiones de las aplicaciones agotan los recursos de la CPU necesarios para la autenticación y la administración de las sesiones.
- Agregaciones complejas: las Multi-stage operaciones (ordenar, agrupar, unir) consumen una cantidad significativa de CPU, especialmente en conjuntos de datos grandes sin índices optimizados.
- Consultas ineficientes: el uso elevado de la CPU se debe al escaneo de grandes conjuntos de resultados y operaciones con expresiones regulares.
- Write-Heavy Cargas de trabajo: las actualizaciones frecuentes y las inserciones masivas agotan la CPU con el procesamiento de documentos, las actualizaciones de índices y el registro de transacciones.

Investigue: recopile métricas

Amazon DocumentDB proporciona supervisión a través de Amazon. CloudWatch Las métricas de Amazon DocumentDB se pueden clasificar en términos generales en métricas a nivel de clúster y a nivel de instancia. Para obtener información sobre los recursos relacionados con la CPU, la memoria y las conexiones, consulte las métricas a nivel de instancia, ya que el consumo de recursos es específico de la instancia. Identifica las instancias (principales o réplicas) que muestran los picos de CPU y anota los momentos en que se producen los picos.

La CloudWatch métrica del uso de la CPU para`CPUUtilization`.

Diagnosticar: encontrar la causa raíz

Tras identificar la instancia y el momento en que se produjo el pico de CPU, lo siguiente es diagnosticar la causa que está afectando en gran medida al uso de la CPU. Como hay varios motivos que pueden provocar un pico de CPU, identifiquemos qué es lo que contribuye a ello, empezando por:

Conexiones: compruebe CloudWatch las métricas `DatabaseConnections` y`DatabaseConnectionsMax`, si hay picos en las conexiones y coinciden con el tiempo de inactividad de la CPU, es probable que el aumento de las conexiones durante un período breve sea la causa del pico de la CPU.

Consultas: si no es la conexión, la causa podría ser una consulta. Si has activado Performance Insights, ve a la consola de Performance Insights de la instancia y consulta las consultas de la instancia. La otra cosa a la que hay que prestar atención a las consultas es la del generador de perfiles.

Resolver: solucione el problema

Ráfagas de conexión: si las ráfagas de conexión son la causa de los picos de CPU, considere la posibilidad de reutilizar las conexiones en su aplicación o implementar la agrupación de conexiones. Para obtener más información sobre cómo optimizar las conexiones altas, consulte. the section called “Problemas de conectividad”

Distribución de consultas: si los picos de CPU se deben a consultas de lectura en la instancia principal, considere la posibilidad de redirigir las operaciones de lectura a las instancias de réplica si los resultados de la consulta finalmente pueden ser coherentes.

Optimizaciones de consultas: si una consulta o un conjunto de consultas provocan un aumento de la CPU, considera la posibilidad de optimizar la consulta, por ejemplo, añadiendo un índice si la consulta no lo utiliza. Para obtener más información sobre cómo optimizar una consulta, consultethe section called “La consulta se ejecuta lentamente”.

Escalado de instancias: si el uso de la CPU sigue siendo alto y constante después de los esfuerzos de optimización, actualice a una clase de instancia más grande con más capacidad de CPU para gestionar sus requisitos de carga de trabajo de forma eficaz.

Sin servidor: en el caso de cargas de trabajo impredecibles que provoquen picos de CPU de corta duración, considere la posibilidad de utilizarla. Uso de Amazon DocumentDB sin servidor Las

instancias sin servidor escalan automáticamente los recursos en función de la demanda, lo que elimina la necesidad de aprovisionar tamaños de instancia fijos para dichas cargas de trabajo.

Problemas de conexión con Amazon DocumentDB

Identificación: detecte el problema

Causas comunes

Los problemas de conexión suelen deberse a tres áreas principales:

El agotamiento del grupo de conexiones se produce cuando una aplicación alcanza el número máximo de conexiones permitido a Amazon DocumentDB, ya sea a través de los límites del grupo de conexiones del lado del cliente o de las instancias del lado del servidor. Esta condición provoca una degradación del rendimiento de las aplicaciones, tiempos de espera y posibles errores, ya que las nuevas solicitudes de conexión se ponen en cola o se rechazan.

La sobrecarga de autenticación se produce cuando Amazon DocumentDB recibe un exceso de solicitudes de autenticación simultáneas, especialmente cuando procesa más de 1000 conexiones nuevas en un período de tiempo corto. Durante el mantenimiento de la autenticación, Amazon DocumentDB mantiene un bloqueo exclusivo en el mapa de sesiones, lo que hace que los intentos de autenticación posteriores queden en cola hasta que finalice el mantenimiento.

Los problemas de configuración en Amazon DocumentDB suelen deberse a errores de configuración en la red, la seguridad y los ajustes del cliente. Esto incluye elementos como la configuración incorrecta del grupo de seguridad, la configuración incorrecta de la VPC o SSL/TLS los problemas con los certificados. Comprender la configuración adecuada es fundamental para mantener un acceso seguro y fiable a la base de datos.

Diagnosticar: encontrar la causa raíz

Grupos de conexiones

El grupo de conexiones se inicializa al crear una MongoClient instancia. Cada grupo mantiene las conexiones en función de dos parámetros clave:

minPoolSize : cantidad mínima de conexiones mantenidas

maxPoolSize : número máximo de conexiones permitidas

Cuando una solicitud necesita una conexión:

1. El grupo comprueba si hay conexiones inactivas disponibles
2. Si no existe ninguna y el tamaño del grupo es inferior al máximoPoolSize, crea una nueva conexión
3. Si es máximoPoolSize, la solicitud entra en una cola de espera
4. Si la cola está llena o se ha agotado el tiempo de espera, se lanza MongoWaitQueueFullException

El comportamiento de la cola de espera se controla mediante estos parámetros:

wait QueueTimeout MS: tiempo máximo de espera para la conexión

esperarQueueSize : número máximo de solicitudes en cola

Este es un ejemplo de un enfoque problemático para conectarse a Amazon DocumentDB, donde se crea un grupo nuevo cada vez:

```
for(Request request : requests) {
    MongoClient client = MongoClients.create(settings);
    // Process request
    client.close();
}
```

CloudWatch Las métricas críticas que se deben monitorear son:

- `DatabaseConnections`- El número de conexiones (activas e inactivas) abiertas en una instancia tomada con una frecuencia de 1 minuto.
- `DatabaseConnectionsMax`- El número máximo de conexiones de base de datos abiertas (activas e inactivas) en una instancia en un período de 1 minuto.
- `DatabaseConnectionsLimit`- El número máximo de conexiones simultáneas a bases de datos (activas e inactivas) que se permite en una instancia en un momento dado.
- `LowMemNumOperationsThrottled`- El número de solicitudes que se limitan debido a la falta de memoria disponible en un período de 1 minuto.

Para ver las cuotas por clase de instancia, consulte. Cuotas de instancias

Entre las señales de advertencia más comunes que indican problemas con los grupos de conexiones a nivel de aplicación se incluyen las siguientes:

- Incrementar los tiempos de adquisición de conexiones
- Aumento del tamaño de las colas de espera
- Aumenta el número de excepciones por tiempo de espera

Sobrecarga de autenticación

La conexión a Amazon DocumentDB sigue un flujo similar al siguiente:

Solicitud de conexión → SSL Handshake → Autenticación → Creación de sesión → Conexión lista

Cuando se procesen más de 1000 conexiones nuevas, las solicitudes de conexión adicionales pasarán a formar parte de una cola de autenticación tras completar el protocolo de enlace SSL. El tiempo medio de conexión desde tu aplicación aumentará durante estos eventos de sobrecarga.

CloudWatch Las métricas críticas que se deben supervisar son las siguientes:

- `DatabaseConnections`- El número de conexiones (activas e inactivas) abiertas en una instancia tomada con una frecuencia de 1 minuto.
- `DatabaseConnectionsMax`- El número máximo de conexiones de base de datos abiertas (activas e inactivas) en una instancia en un período de 1 minuto.
- `DatabaseConnectionsLimit`- El número máximo de conexiones simultáneas a bases de datos (activas e inactivas) que se permite en una instancia en un momento dado.

Problemas de configuración

El problema de configuración más común se produce cuando se intenta conectarse a un punto final de clúster privado de Amazon DocumentDB desde un entorno sin acceso al entorno de red privada. Amazon DocumentDB solo está diseñado para una nube privada virtual (VPC) y actualmente no admite puntos de conexión públicos. No puede conectarse directamente a su clúster de Amazon DocumentDB desde su portátil o un entorno de desarrollo local fuera de su VPC.

Esto se manifestará en errores como los siguientes:

```
Error: couldn't connect to server...
Failed to connect to...
```

```
exception: connect failed
connection attempt failed
```

Las configuraciones incorrectas de los grupos de seguridad también pueden provocar errores de conexión. De forma predeterminada, un clúster de Amazon DocumentDB detecta las conexiones en el puerto TCP 27017. La aplicación fallará si intenta conectarse a un puerto diferente del puerto con el que se implementó el clúster o si la aplicación no está incluida en la configuración del grupo de seguridad de entrada del clúster.

La administración incorrecta de los certificados también puede provocar problemas de conexión. De forma predeterminada, el cifrado en tránsito está habilitado para los clústeres de Amazon DocumentDB recién creados. Cuando se habilita el cifrado en tránsito, se requieren conexiones seguras mediante TLS para conectarse al clúster mediante el certificado global-bundle.pem. Si intenta utilizar un certificado incorrecto, recibirá errores como los siguientes:

```
unable to get local issuer certificate
```

Si intenta conectarse a un clúster con TLS habilitado sin especificar los parámetros de TLS, recibirá errores como los siguientes:

```
Server selection timed out after 30000 ms
```

Resolver: soluciona el problema

Grupos de conexiones: revise la agrupación de conexiones implementando o ajustando los tamaños de los grupos para que coincidan con los requisitos de carga de trabajo. Las configuraciones óptimas de los grupos dependen de la carga de trabajo y los requisitos. Debe mantener un mínimo PoolSize tal que las conexiones principales estén listas y disponibles, y un máximo lo suficientemente WaitTime corto como para que fallen rápidamente si el grupo se ha agotado.

Este es un ejemplo de cómo reutilizar un único grupo sin crear uno nuevo cada vez:

```
MongoClient client = MongoClients.create(settings);
    for(Request request : requests) {
    // Process request
}
```

Sobrecarga de autenticación: administre la autenticación implementando un aumento gradual de las conexiones y limitando las conexiones nuevas a 1000 por vez. Utilice la agrupación de conexiones para reutilizar las conexiones autenticadas de forma eficaz. Para evitar sobrecargar el clúster de Amazon DocumentDB con conexiones, implemente una estrategia de aumento de las conexiones.

```
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;
        }
    }
}
```

También puede configurar los ajustes de su grupo de conexiones para limitar la cantidad total de conexiones permitidas.

```
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();
```

Problemas de configuración: asegúrese de que la aplicación tenga acceso a la VPC privada y a la subred en las que se encuentran los recursos de Amazon DocumentDB. Si utiliza la interconexión de VPC, consulte la guía para desarrolladores Cómo solucionar problemas de conexión de

interconexión de VPC para obtener más información. También puedes consultar el artículo del Knowledge Center ¿Cómo soluciono los problemas de conectividad desde Internet a las instancias de Amazon EC2 de mi VPC? .

Para la configuración del grupo de seguridad, debe incluir una regla de ingreso en su grupo de seguridad de Amazon DocumentDB para permitir las conexiones desde su aplicación.

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

Si el clúster está configurado con el cifrado TLS, descargue el certificado TLS para Amazon Amazon DocumentDB denominado global-bundle.pem y utilícelo cuando se conecte al clúster.

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

Long-term Soluciones

Es posible que sea necesario escalar las instancias actualizándolas a una clase de instancia más grande o añadiendo réplicas de lectura para distribuir la carga de conexión. La implementación adecuada del equilibrio de carga garantiza una utilización óptima de los recursos en todo el clúster.

Los cambios en las aplicaciones deben centrarse en implementar una gestión sólida de las conexiones, una supervisión integral y el cumplimiento de las mejores prácticas de agrupación de

conexiones. Esto incluye el manejo adecuado de los errores y la administración del ciclo de vida de las conexiones.

Las mejoras en la arquitectura pueden incluir la adopción de Amazon DocumentDB Serverless para cargas de trabajo variables, la implementación de una lógica de reintento sofisticada y el diseño para que sea tolerante a errores. Considere la posibilidad de reestructurar la arquitectura de las aplicaciones para gestionar mejor la administración de conexiones.

Prácticas recomendadas

Grupos de conexiones

Mediante la administración y la supervisión adecuadas de los grupos de conexiones, las aplicaciones pueden mantener una conectividad estable con las bases de datos y, al mismo tiempo, evitar situaciones de agotamiento que podrían afectar a la confiabilidad y el rendimiento del sistema. Configure los tiempos de espera adecuados y defina el tamaño de su grupo en función de las características de su carga de trabajo.

Ejemplo de configuración de un grupo de conexiones

```
MongoClientSettings settings = MongoClientSettings.builder()
    .applyToConnectionPoolSettings(builder ->
        builder.maxSize(10))
    .applyToConnectionPoolSettings(builder ->
        builder.maxWaitQueueSize(2))
    .applyToConnectionPoolSettings(builder ->
        builder.maxConnectionIdleTime(10, TimeUnit.MINUTES))
    .build();
```

Para obtener más información, consulte Creación de aplicaciones resilientes con Amazon DocumentDB, parte 1: Configuración del cliente.

Sobrecarga de autenticación

Implemente siempre la agrupación de conexiones con los valores adecuados para los parámetros en función de su carga de trabajo. Utilice una técnica de establecimiento de conexiones gradual y mantenga las conexiones persistentes, siempre que sea posible. Implemente una limpieza adecuada de las conexiones para garantizar que no se desperdicien recursos inactivos.

Problemas de configuración

Asegúrese de haber configurado el enrutamiento adecuado desde su aplicación a los recursos de Amazon DocumentDB. Utilice TLS para el cifrado en tránsito e implemente el acceso con privilegios mínimos. Verifique sus credenciales de Amazon DocumentDB y valide los valores de las cadenas de conexión.

La consulta se ejecuta lentamente

Identificación: detecta el problema

Una consulta de ejecución lenta es aquella que supera el tiempo de ejecución esperado para los requisitos empresariales. La definición de lo que constituye una consulta lenta varía según las distintas cargas de trabajo. Puede identificar las consultas lentas mediante los registros del generador de perfiles o Performance Insights.

1. Habilite el generador de perfiles si aún no lo ha hecho. Los registros de Profiler se escriben CloudWatch en Amazon en el grupo de registros:`/aws/docdb/<cluster-identifier>/profiler`. Utilice CloudWatch Logs Insights para consultarlos.

   Ejemplo de consulta de análisis de CloudWatch registros para obtener las 10 consultas más lentas de la colección ecommerce.products:

   ```
   filter ns="ecommerce.products" | sort millis desc | limit 10
   ```

2. Utilice Performance Insights para identificar consultas costosas prácticamente en tiempo real. Activa Performance Insights si aún no está activado.
   a. Abra la AWS consola, vaya a Amazon DocumentDB, luego a Performance Insights y seleccione la instancia de clúster.
   b. Revise el cronograma de carga de base de datos (AAS) y las consultas principales (por carga de base de datos). Amplíe el resumen de una consulta para ver las sentencias secundarias literales.
   c. Capture las consultas que requieren análisis.

   > **Note**
   >
   > Es posible que no todas las consultas de Performance Insights sean consultas ineficientes o lentas.

Investigue: recopile información

1. El generador de perfiles proporciona el plan de ejecución de la consulta y las métricas clave asociadas a él, como millis, nreturned y planSummary (índice de uso):


```
{
    "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"
}
```

Para buscar las consultas de COLLSCAN:

```
filter planSummary="COLLSCAN" | sort millis desc | limit 20
```

2. Utilice Performance Insights para analizar las tendencias de ejecución de consultas, como las esperas, la carga y el impacto en los recursos (p. ej., IOPS o CPU por consulta) en tiempo real.

   Como Performance Insights no proporciona el plan de ejecución de la consulta, capture la consulta y ejecute explain («ExecutionStats») en la consulta en su clúster de Amazon DocumentDB:

   ```
   db.ecommerce.products.find().explain("executionStats")
   ```

3. Si lo desea, correlacione las métricas del generador de perfiles con los datos de Performance Insights (por ejemplo, compare las consultas de alto número de milisegundos del generador de perfiles con las consultas principales de Performance Insights).

Diagnostique: encuentre la causa raíz

En este paso, diagnostique el plan de consulta para identificar posibles optimizaciones. Utilice el siguiente flujo: síntoma → causa probable:

- Síntoma: Resumen del plan: «COLLSCAN»

  Causa: falta el índice o es incorrecto.

- Síntoma: la agregación es lenta con $group/$sort

  Causa: es posible que la canalización de agregación esté procesando demasiados datos en la memoria.

- Síntoma: latencias altas, mientras que Performance Insights muestra la carga de base de datos agrupada por esperas de E/S, aunque se utilizan índices (PlanSummary: «IXSCAN»).

  Causa: consultas I/O enlazadas; los índices o el conjunto de trabajo superan la caché de búfer disponible en la instancia.
- Síntoma: el PI muestra los estados de espera de la CPU, mientras que el AAS es alto debido a las pocas consultas

  Causa: CPU-bound consultas (agregaciones complejas, $regex).
- Síntoma: muchas consultas lentas, pero ninguna muestra un PlanSummary caro

  Causa: cuello de botella externo (red, limitaciones, tareas de mantenimiento, volumen de consultas).

Resolver: solucione el problema

Cuando implemente correcciones, valide siempre con explain («ExecutionStats») y supervise la carga de la base de datos Performance Insights.

1. Resumen del plan: «COLLSCAN»
   - Cree un índice específico.

     Ejemplo: para consultas frecuentes que filtran por {categoría, precio} y ordenan por precio descendente:

     ```
     db.products.createIndex({ category: 1, price: -1 })
     ```
2. la agregación es lenta con $group/$sort
   - Empuje $match y $project con antelación para reducir el flujo de documentos hacia $group/$sort.
   - Limite el número de campos al principio del proceso:

     ```
     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. latencias altas, mientras que Performance Insights muestra la carga de base de datos agrupada por esperas de E/S
   - Valide si BufferCacheHitRatio el nivel de ReadiOps de la instancia es bajo o alto. Aumente la memoria de la instancia (actualice la clase de instancia, por ejemplo, r6g.large → r6g.xlarge) o utilice la clase de instancia NVMe.
   - Reduzca la huella de índice.
   - Agregue réplicas de lectura para reducir el tráfico de lectura (utilice la configuración ReadPreference para redirigir el tráfico de las réplicas si la consulta tolera la coherencia final).
4. El PI muestra los estados de espera de la CPU, mientras que el AAS es alto debido a las pocas consultas
   - Sustituya la costosa opción $regex por búsquedas de prefijos indexados o por el índice $text.
   - Escritura por lotes (inserción) para reducir la amplificación de la escritura.

> **Note**
>
> Pruebe siempre los cambios en un entorno inferior (no de producción) antes de promocionarlos a producción.

Anti-patterns

En esta sección se describen los antipatrones más comunes que pueden afectar negativamente al rendimiento de Amazon DocumentDB.

Temas

- Índice compuesto con más de 3 atributos
- Consultas de larga duración
- Índices poco utilizados y redundantes
- Multi-Key Índices con matrices grandes

Índice compuesto con más de 3 atributos

Descripción general de

Un índice compuesto o compuesto mantiene las referencias a varios campos dentro de una única estructura de índice. Estos índices optimizan el rendimiento de las consultas que filtran varios campos simultáneamente o combinan el filtrado con las operaciones de clasificación. También son eficaces para las consultas de una sola condición en los campos indexados situados más a la izquierda. La base de datos aprovecha estas entradas del índice para localizar de forma eficiente los documentos coincidentes sin necesidad de digitalizar toda la colección.

Amazon DocumentDB puede usar índices compuestos para admitir consultas que incluyan cualquier subconjunto principal de los campos indexados, un concepto conocido como prefijos de índice. Por ejemplo, si tiene activado un índice compuesto`{state: 1, city: 1, zipcode: 1}`, Amazon DocumentDB puede procesar eficazmente consultas que utilicen únicamente el campo «estado», los campos «estado» y «ciudad» juntos o los tres campos «estado», «ciudad» y «código postal». Sin embargo, las consultas deben utilizar los campos de izquierda a derecha sin omitir ningún campo intermedio. Esto significa que las consultas que utilizan únicamente los campos «ciudad» o «código postal», o combinaciones como «estado» y «código postal» (omitiendo «ciudad»), no pueden utilizar completamente el índice.

En la mayoría de los escenarios del mundo real, los índices compuestos con tres o menos atributos suelen lograr un rendimiento y una eficiencia de recursos óptimos. Si bien es posible tener más de 3 atributos en el índice compuesto, a menudo se traduce en un mayor consumo de recursos, lo que compensa los beneficios.

Impacto en el clúster

Si bien resulta tentador crear un índice que abarque todas las condiciones de una consulta para lograr el máximo rendimiento teórico, la mayor parte del filtrado de datos se realiza a través de los primeros 1 a 3 atributos de un índice compuesto. Los campos adicionales que superen este umbral contribuyen principalmente al tamaño del índice más que a una optimización significativa de las consultas.

Almacenamiento y I/O gastos generales: los índices compuestos con muchos atributos consumen mucho más espacio de almacenamiento que las alternativas más sencillas. El tamaño es directamente proporcional al número de atributos indexados y al tamaño de los propios valores indexados.

Consumo de memoria: el gran espacio de almacenamiento de los índices compuestos con muchos atributos crea un espacio correspondiente en la memoria, lo que hace que el conjunto de trabajo sea más grande y desplaza otros datos a los que se accede con frecuencia del conjunto de búferes.

Operaciones de escritura: cada modificación de un documento que afecte a varios campos indexados requiere actualizar toda la entrada del índice compuesto, lo que multiplica el trabajo necesario para completar la operación de escritura.

¿Cómo identificarlo

Comience por revisar todos los índices de su colección para identificar los índices compuestos con más de tres atributos:

```
// 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 }
```

Corrección

Si se utiliza un índice compuesto con más de tres atributos, identifique las consultas que lo utilizan y busque oportunidades de optimización. Considere la posibilidad de reemplazar estos índices por otros más eficientes que contengan tres o menos atributos. Tras implementar los nuevos índices, elimine los índices antiguos que contienen más de tres atributos.

> **Note**
>
> Coordínese siempre con las partes interesadas y valide el impacto en el rendimiento antes de eliminar cualquier índice.

Siga la regla de igualdad, orden y rango (ESR) al crear índices compuestos.

Regla de rango de clasificación por igualdad (ESR): este orden maximiza la eficiencia del índice al filtrar primero el conjunto de datos con condiciones de igualdad, luego aplicar operaciones de

clasificación en el conjunto reducido y, por último, realizar escaneos de rango en el subconjunto más pequeño posible

- ordenar los campos como Igualdad (coincidencias exactas),
- Ordenar (ordenar),
- Rango (>, <, $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
```

Consultas de larga duración

Descripción general de

Long-running las consultas pueden provocar problemas de rendimiento en todo el clúster. Estas consultas pueden interferir con el proceso de recopilación de elementos no utilizados del Control de Multi-Version Concurrencia (MVCC) de Amazon DocumentDB, lo que provoca la acumulación de documentos con versiones antiguas y una degradación del rendimiento en todo el clúster. Amazon DocumentDB implementa un tiempo de espera de 2 horas en el servidor como mecanismo de seguridad para evitar que las consultas incontroladas consuman recursos de forma indefinida.

Qué son las consultas de larga duración:

- Consultas que se ejecutan durante períodos prolongados (normalmente > 30 minutos)
- Abre los cursores que permanecen activos durante horas (no se aplicará un tiempo de espera de 2 horas si el cursor está activo)

Impacto en el clúster

Las consultas de larga duración pueden interferir con el proceso de recolección de basura

- Las versiones antiguas se acumulan: el recolector de basura no puede recuperar versiones antiguas de documentos
- Abundamiento de colecciones e índices: las entradas de recopilación e índice se acumulan con el tiempo, lo que aumenta y, por lo tanto, puede resultar en un mayor costo de almacenamiento.
- Presión de la CPU y la memoria: la presión de la CPU y la memoria aumenta debido al procesamiento ineficiente del mayor número de versiones antiguas de documentos, entradas de índice e ID de transacciones.

Consulta de larga duración → Bloquea el GC → Crecimiento del almacenamiento → Presión de la CPU y la memoria → Consultas más largas

Supervise y detecte

1. Para buscar consultas de ejecución prolongada, utilice el 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. Para buscar cursores que estén activos durante más de 30 minutos

```
// 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. Monitorear el progreso del recolector de basura mediante CloudWatch

`LongestRunningGCProcess`— Duración en segundos del proceso de recolección de basura activo más prolongado. Se actualiza cada minuto y realiza un seguimiento únicamente de las operaciones activas, excluyendo los procesos que se completan en el intervalo de un minuto.

`AvailableMVCCIds`-Un contador que muestra el número de operaciones de escritura restantes disponibles antes de llegar a cero. Cuando este contador llega a cero, el clúster pasa al modo de solo lectura hasta que se recuperen y reciclen los identificadores. El contador disminuye con cada operación de escritura y aumenta a medida que la recopilación de elementos no utilizados recicla los identificadores de MVCC antiguos.

> **Note**
>
> Los ID de MVCC más bajos y la duración prolongada de la recolección de basura no se atribuyen exclusivamente a consultas de larga duración. Write-intensive Las cargas de trabajo en instancias con recursos limitados también pueden reducir la disponibilidad de los identificadores de MVCC y prolongar los ciclos de recolección de basura.

Estrategias de remediación

- Implemente los tiempos de espera de las consultas en la aplicación
- No mantenga los cursores activos durante períodos más largos
- Optimice las consultas para obtener un mejor rendimiento.
- Prefiera el procesamiento por lotes de las operaciones de escritura

Índices poco utilizados y redundantes

Descripción general de

Los índices mejoran el rendimiento general de las consultas de las cargas de trabajo, pero conllevan costes. Cada índice consume almacenamiento y I/O recursos, y las operaciones de escritura deben actualizar todos los índices de la colección. Si la base de datos contiene índices redundantes, infrautilizados o no utilizados, se generará una sobrecarga para el clúster y se reducirá el rendimiento. Varios índices innecesarios crean un antipatrón que provoca problemas de rendimiento, un aumento de los costes de almacenamiento y una mayor sobrecarga operativa.

Escenarios de indexación subóptimos

- Índices no utilizados: índices creados para consultas únicas o en iteraciones de productos anteriores a las que ya no se accede mediante los patrones de consulta actuales. Para identificar los índices no utilizados de una colección, puede utilizar IndexStats. Para obtener más información, consulte the section called “Análisis del uso de los índices e identificar los índices no utilizados”.
- Índices redundantes: varios índices que cubren patrones clave superpuestos o patrones de consulta idénticos. Ejemplo: un índice de clave única en el atributo A se vuelve redundante cuando un índice compuesto cubre tanto el atributo A como el B, ya que el índice compuesto puede gestionar consultas solo sobre el atributo A.
- Over-Indexing: Crear índices «por si acaso» sin analizar los patrones de consulta reales ni los requisitos de rendimiento.
- Índices de cardinalidad baja: índices en campos con pocos valores distintos (por ejemplo, campos booleanos o indicadores de estado) que proporcionan un beneficio mínimo de optimización de las consultas.

Impacto en el rendimiento del clúster

- Almacenamiento y E/S: cada índice consume espacio de almacenamiento y recursos de E/S, lo que contribuye a los costos generales. Si un índice no se utiliza o es redundante, representa un coste innecesario para el clúster.
- Rendimiento de escritura reducido: las operaciones de inserción, actualización y eliminación deben mantener los índices, lo que genera una sobrecarga que consume recursos. Los índices redundantes o no utilizados añaden una sobrecarga innecesaria a estas operaciones.

- Presión de memoria y CPU: los índices compiten por la memoria del grupo de búferes y pueden desalojar los datos a los que se accede con frecuencia si el conjunto de trabajo no cabe en la memoria de la instancia, lo que resulta en un nivel subóptimo. BufferCacheHitRatio El sistema también consume recursos de la CPU manteniendo las estructuras de índice.

Herramientas disponibles para identificar las oportunidades de optimización de índices

- Herramienta de revisión de índices:
- Herramienta de detección de cardinalidad de índices

Recopilación del kit de herramientas:

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

1. Herramienta de revisión de índices

La herramienta de revisión de índices analiza todas las colecciones e índices para proporcionar una lista de los índices y sus patrones de uso e identifica los índices redundantes en el clúster de Amazon DocumentDB. Ejecute esta herramienta en todas las instancias del clúster para realizar un análisis exhaustivo. Para obtener más información, consulte la herramienta de revisión de índices.

Uso:

```
cd performance/index-review/

python3 index-review.py --server-alias <server-alias> --uri <mongodb-uri>
```

Archivos de salida

- `{server-alias}-collections.csv`: Collection-level estadísticas
- `{server-alias}-indexes.csv`: Métricas detalladas de uso del índice
- `{server-alias}-{timestamp}-index-review.json`: Datos sin procesar para su posterior análisis

> **Note**
>
> Ejecute para cada instancia del clúster con un alias independiente para generar estadísticas de cada instancia y, a continuación, ejecute la herramienta para obtener un análisis acumulativo

Por ejemplo: `python3 index-review.py --files "{server-alias}-{timestamp}-index-review.json, {server-alias}-{timestamp}-index-review.json" --server-alias full-review`

2. Herramienta de detección de cardinalidad de índices

Esta herramienta identifica los índices con baja cardinalidad que son ineficientes para el rendimiento de las consultas.

Uso

```
cd performance/index-cardinality-detection/

python3 detect-cardinality.py --uri <mongodb-uri>
```

Parámetros

- `--threshold`: porcentaje de umbral de cardinalidad (predeterminado: 1%)
- `--sample-count`: Número de documentos para muestrear (predeterminado: 100.000)
- `--max-collections`: Número máximo de colecciones que se pueden escanear por base de datos (predeterminado: 100)

Para obtener más información, consulte https://aws.amazon.com/blogs/database/detect-and-fix-low-cardinality-indexes-in-amazon-documentdb/

Estrategias de remediación

1. Eliminación de índices redundantes y no utilizados

```
// Example: Drop unused index
```

```
db.collection.dropIndex("unused_index_name")
```

> **Note**
>
> Los índices nunca deben descartarse sin consultarlos con todas las partes interesadas y comprobar su rendimiento.

2. Optimización del índice de baja cardinalidad

- Utilice índices parciales con filtros
- Convierta índices únicos de baja cardinalidad en índices compuestos

Multi-Key Índices con matrices grandes

Descripción general de

Multi-key los índices de Amazon DocumentDB le permiten consultar campos de matriz de forma eficaz. Esta estrategia de indexación ofrece latencias de consulta significativamente más bajas cuando se busca dentro de una matriz de datos, lo que la convierte en una opción atractiva para aplicaciones con estructuras de datos complejas.

Sin embargo, cuando se trabaja con matrices grandes o con varios campos de matrices dentro de una colección, es importante comprender las implicaciones en cuanto a los recursos y las características de rendimiento para poder tomar decisiones arquitectónicas informadas. Al crear un índice en un campo de matriz, Amazon DocumentDB genera entradas de índice individuales para cada elemento de la matriz.

Impacto en el clúster

- Sobrecarga de almacenamiento y E/S: Multi-key los índices a veces pueden ocupar espacio de almacenamiento varias veces más que el de la tabla base; el tamaño es directamente proporcional al de los documentos de la colección con el atributo de matriz indexada, el número de elementos de la matriz y el tamaño del elemento.
- Uso de memoria: dado que el espacio de almacenamiento es relativamente grande, estos índices tienen el mismo espacio en la memoria, lo que da como resultado un conjunto de trabajo más grande.

- Operaciones de escritura: Multi-key los índices generan una sobrecarga durante las operaciones de escritura. Cada elemento de la matriz genera entradas de índice independientes, lo que multiplica el trabajo necesario para completar la operación de escritura.

Estrategias de remediación

- Cree un índice de varias claves si es necesario y limite el número de campos de la matriz.
- Limite el número de índices de varias claves en la colección
- Considere la posibilidad de modificar su modelo de datos

Consejos para mejorar el rendimiento

En esta sección se proporcionan cinco estrategias de optimización del rendimiento para Amazon DocumentDB a fin de mejorar la eficiencia de las aplicaciones y la ejecución de consultas.

1. Utilice $match como primera etapa en las canalizaciones de agregación

Coloca siempre $match como la primera etapa de filtrado de tu canalización de agregación para maximizar el rendimiento. Amazon DocumentDB utilizará los índices de forma eficaz cuando $match lidere la canalización, lo que permitirá a la base de datos filtrar los datos de forma temprana y reducir la sobrecarga de procesamiento.

```
// Optimized approach
db.orders.aggregate([
  { $match: { status: "active", category: "electronics" } }, // Index utilization
  { $group: { _id: "$category", total: { $sum: "$price" } } },
  { $sort: { total: -1 } }
])
```

Impacto: el filtrado temprano reduce la cantidad de documentos procesados en las etapas posteriores del proceso, lo que se traduce en una ejecución más rápida de las consultas y un menor consumo de recursos.

2. Utilice $project en Aggregation Pipeline para minimizar el tamaño de los datos de la canalización

Realice solo los campos esenciales en las etapas de su proceso de agregación para minimizar el tamaño de los datos y mejorar el rendimiento. Utilice $project de forma estratégica para incluir solo los datos que necesita.

```
// 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" } } }
])
```

Impacto: Los documentos más pequeños reducen el uso de memoria y mejoran la eficiencia del procesamiento de los procesos, lo que se traduce en un mejor rendimiento general de las consultas.

3. Habilite la compresión de documentos para reducir los costes de almacenamiento y mejorar el rendimiento de las consultas I/O

Habilite la compresión de documentos desde el grupo de parámetros del clúster para reducir los costos y I/O los costos de almacenamiento y aumentar el rendimiento de las consultas. Amazon DocumentDB almacena los documentos comprimidos en el disco y en la RAM, lo que reduce el consumo de memoria y I/O los costes.

Impacto:

- Caben más documentos en la memoria disponible
- Acceso más rápido a los datos con lecturas de disco reducidas
- Reduzca los costes y I/O costes de almacenamiento y mejore el rendimiento de las consultas

> **Note**
>
> Amazon DocumentDB no habilita la compresión de forma predeterminada para la versión 5.0. Puede habilitar la compresión a nivel de colección o clúster para el clúster 5.0. Utilice

la utilidad de revisión de compresión de Amazon DocumentDB para analizar las tasas de compresión de sus colecciones.
En Amazon DocumentDB 8.0, la compresión está habilitada de forma predeterminada.

4. Aproveche los índices para obtener un rendimiento óptimo de las consultas

Asegúrese de que sus consultas utilicen siempre índices para obtener un rendimiento óptimo. Amazon DocumentDB ofrece varios tipos de índices para adaptarse a diferentes casos de uso.

Principios de indexación:

- Cada consulta debe aprovechar un índice adecuado
- Amazon DocumentDB proporciona varios tipos de índices
- Los índices compuestos ofrecen la mayor flexibilidad al admitir varias formas de consulta con un único índice
- Diseñe índices para admitir simultáneamente las operaciones de clasificación y filtrado

Descripción de los prefijos de índice: los índices compuestos funcionan con prefijos de índice; Amazon DocumentDB puede utilizar cualquier subconjunto de campos de índice de izquierda a derecha. Por ejemplo, el índice crea estos prefijos utilizables: `{ category: 1, price: -1, inStock: 1 }`

- `{ category: 1 }`- admite el filtrado de consultas únicamente por categoría
- `{ category: 1, price: -1 }`- admite el filtrado de consultas por categoría y sorting/filtering por precio
- `{ category: 1, price: -1, inStock: 1 }`- admite la consulta compuesta completa

Las consultas sobre precio, InStock o InStock por sí solas no utilizarán este índice, ya que no comienzan por el primer campo (categoría).

Cómo identificar las consultas que no utilizan índices: utilice el método explain () para analizar la ejecución de las consultas e identificar las consultas que escanean las colecciones en lugar de utilizar índices.

Impacto: las consultas sin utilizar los índices dan como resultado escaneos de colecciones, lo que aumenta la presión de la memoria y la CPU sobre la instancia y aumenta la latencia de las consultas.

5. Optimice los modelos de datos en función de los patrones de consulta

Alinee su modelo de datos con la forma en que su aplicación consulta y actualiza los datos. El modelado de datos es la base de las aplicaciones Amazon DocumentDB de alto rendimiento.

Estrategias de optimización:

Integración para el rendimiento

- Almacene los datos relacionados juntos cuando se acceda a ellos con frecuencia como una unidad
- Inserte juntos los documentos que siempre se recuperan
- Adecuado para relaciones de uno a pocos

```
// 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" }
  ]
}
```

Haciendo referencia a la flexibilidad

- Utilice referencias para datos grandes o a los que se accede con poca frecuencia
- Se recomienda para relaciones de uno a varios con conjuntos de datos grandes
- Evita la sobrecarga de documentos y mejora el rendimiento de las actualizaciones

Estrategia de división de colecciones

Cuando solo unos pocos campos de documentos grandes se actualizan con frecuencia, o cuando los documentos están llenos de datos de gran tamaño a los que se accede con poca frecuencia, considere la posibilidad de dividir las colecciones:

- Mantenga los campos que se actualizan con frecuencia en una colección separada y más pequeña
- Guarde los datos estáticos o a los que se accede con poca frecuencia en otra colección
- Conéctelos con referencias cuando sea necesario

```
// 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 de rendimiento: los documentos más pequeños se traducen en actualizaciones más rápidas, un menor uso de memoria y una mayor eficiencia de la memoria caché.

Impacto: el modelado de datos ineficiente provoca consultas subóptimas, un aumento del tamaño de los documentos y un aumento del uso de memoria, lo que reduce el rendimiento de las aplicaciones y aumenta los costes operativos.

Análisis del plan de consultas

El análisis del plan de consultas mediante la explicación del plan proporciona información esencial sobre el rendimiento de las consultas en Amazon DocumentDB. El plan de consultas con ExecutionStats revela métricas clave que incluyen:

- Documentos devueltos por etapa (nReturned)
- Stage-specific tiempos de ejecución (ejecuciónTimeMillisEstimate)
- Duración de la generación del plan (planificaciónTimeMillis)

Al examinar el resultado del plan de consultas, los desarrolladores pueden analizar los patrones de ejecución, evaluar la utilización de los índices e identificar las posibles oportunidades de optimización en todas las etapas del proceso de consulta.

Para analizar el plan de consultas, puede usar el comando explain () en los siguientes formatos.

```
db.runCommand({explain: {query document}, verbosity: "executionStats"})
db.collection.find().explain("executionStats");
```

A continuación se muestra un ejemplo de operación:

```
db.collection.find({ companyname: { '$eq': 'ANYCOMPANY' }, isDeleted: { '$eq':
 false } }).sort({"createdAt":1}).limit(2).explain("executionStats");
```

La salida de esta operación será similar a lo que se indica a continuación:

```
{
  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 })
}
```

A continuación, se incluye un análisis detallado del plan de ejecución de consultas de Amazon DocumentDB, en el que se desglosan cada componente y sus características de rendimiento.

Plazo general

la ejecución TimeMillis representa el tiempo total que tarda la consulta, incluido el tiempo de planificación.

la planificación TimeMillis representa el tiempo total de planificación que tarda la consulta.

Etapas de ejecución

Describe el proceso paso a paso que Amazon DocumentDB utiliza para ejecutar una consulta y muestra cómo fluyen los datos a través de diferentes operaciones.

```
"executionStages": {
    "stage": "[STAGE_NAME]",
    "nReturned": "[NUMBER_OF_DOCS]",
    "executionTimeMillisEstimate": "[TIME]",
    "inputStage": {
        // Nested stages
    }
}
```

Etapas comunes de un plan de consultas

A continuación se muestran las etapas de ejecución habituales de un plan de consultas. Cada etapa devuelve métricas de ejecución TimeMillisEstimate (tiempo de ejecución) y nReturned (cantidad de documentos) para ayudar a evaluar el rendimiento de las consultas en cada etapa.

COLLSCAN (escaneo de colecciones)

- Escanea toda la colección, documento por documento
- nReturned: devuelve todos los documentos coincidentes de la colección
- Puede ser una operación costosa, vista cuando no hay un índice disponible

IXSCAN (escaneo de índices)

- Utiliza un índice para buscar documentos coincidentes
- nReturned: solo los documentos coincidentes según el índice
- Funcionamiento eficiente, preferido a COLLSCAN

SORT

- Ordena los documentos en función de los campos especificados
- Si el atributo de ordenación de la consulta no forma parte de un índice, esta etapa aparecerá de forma explícita
- nReturned: el mismo número que los documentos de entrada
- Memory-intensive para conjuntos de resultados grandes. Para optimizarlo, añada el nombre del campo de ordenación a un índice

LIMIT_SKIP

- Controla el número de documentos devueltos y omitidos
- nReturnado: limitado por el valor LÍMITE
- Se utiliza para operaciones de paginación o LIMIT

SUBESCANEO

- Realiza operaciones de consulta anidadas
- nReturned: varía según los resultados de la subconsulta
- Se utiliza en consultas complejas con múltiples etapas

Las etapas con una alta ejecución TimeMillisEstimate son buenas candidatas para la optimización.

> **Note**
>
> El parámetro ExecutionStats no admite actualmente los comandos de actualización y eliminación.

Comprensión de los documentos examinados en Explicar las estadísticas de ejecución del plan

Cuando ejecuta una consulta`explain("executionStats")`, Amazon DocumentDB proporciona métricas de examen que le ayudan a comprender cuántos documentos se escanearon para generar los resultados de la consulta. Estas métricas son útiles para identificar planes de consultas ineficientes y optimizar el rendimiento.

> **Note**
>
> Disponible solo en Amazon DocumentDB 5.0.1+ y 8.0.0+.

Campos

Campo	Description (Descripción)	Nivel
total DocsExamined	Número total de documentos examinados en todas las etapas de ejecución.	Top-level Execution Stats
Documentos examinados	Número de documentos examinados en una etapa de ejecución específica.	Stage-level

El `docsExamined` campo aparece en las siguientes etapas:

Etapa	Description (Descripción)
COLLSCAN	Escaneo de colecciones. Se examinan todos los documentos de la colección.
IXSCAN	Escaneo de índices. Solo se examinan los documentos que coinciden con la condición del índice.
FETCH	Cuando el optimizador recupera los documentos en una etapa separada de la del IXSCAN, la etapa FETCH informa sobre DocsExamined. En los planes de digitalización indexada, las etapas secundarias del IXSCAN no incluyen DocsExamined. En los planes $lookup, tanto la etapa FETCH como sus etapas secundarias pueden incluir DocsExamined.

> **Note**
>
> `docsExamined`no aparece en las etapas de IXONLYSCAN porque la consulta se resuelve en su totalidad desde el índice sin acceder a los documentos.

Mostrar los documentos examinados en el resultado de Explain Plan ExecutionStats

En los ejemplos siguientes se utiliza una colección de 500 000 documentos y se muestra cómo `docsExamined` cambia en las distintas etapas de ejecución de las consultas.

Cree datos de muestra:

```
for (let i = 0; i < 500000; i++) {
    db.coll.insertOne({ number: i, arr: [i, [i+1]], value: "test", bool: i % 2 === 
 0 });
}
```

Escaneo de colecciones (COLLSCAN)

Sin un índice, Amazon DocumentDB realiza un escaneo completo de la colección.

```
db.coll.find({ "number": { "$lt": 500 } }).explain("executionStats").executionStats
```

Salida:

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

Se examinaron los 500 000 documentos para obtener 500 resultados. La creación de un índice en el campo numérico mejora esta situación.

Escaneo de índices (IXSCAN)

```
db.coll.createIndex({ number: 1 });
db.coll.find({ "number": { "$lt": 5000 } }).explain("executionStats").executionStats
```

Salida:

```
{
    "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 el índice, solo se examinaron y buscaron 5000 de los 500 000 documentos, lo que coincidió con el número devuelto. La condición de índice filtró 495 000 documentos que no coincidían con la consulta.

Escaneo de índices con filtro residual

```
db.coll.find({ "number": { "$lt": 5000 }, "arr": { "$gt":
 4000 } }).explain("executionStats").executionStats
```

Salida:

```
{
    "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 condición del índice coincidió con 5000 documentos de un total de 500 000 y, a continuación, el filtro residual activado `arr` redujo el resultado a 999. El `docsExamined` valor de 5000 refleja todos los documentos examinados después de la condición de indexación, pero antes de aplicar el filtro residual.

Realice la búsqueda con IXSCAN

```
db.coll.find({ "$or": [{ "number": { "$lt": 100000 } }, { "number": { "$gt":
 400000 } }] }).explain("executionStats").executionStats
```

Salida:

```
{
```

```
    "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
                                }
                            }
                        ]
                    }
```

```
                }
            ]
        }
    }
}
```

Cuando el optimizador de Amazon DocumentDB utiliza una etapa de recuperación para recuperar documentos, la etapa FETCH informa. `docsExamined` Las etapas secundarias de IXSCAN no generan informes `docsExamined` porque solo escanean claves de índice sin acceder directamente a los documentos.

FETCH con búsqueda agregada

```
db.coll.explain("executionStats").aggregate([
    { $match: { "number": { "$lt": 5 } } },
    { $lookup: { from: "coll", pipeline: [{ $match: { "number": { "$lt": 3 } } }], as:
 "sub" } }
]).executionStats
```

Salida:

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

El IXSCAN externo examinó 5 documentos cuyo número era inferior a 5. El IXSCAN interno examinó 3 documentos que coincidían con un número inferior a 3 y la etapa FETCH examinó 1 resultado agregado. El `totalDocsExamined` número 9 es la suma de todas las etapas (5 + 3 + 1).

Index-only escaneos (consultas cubiertas)

Un escaneo solo de índices (también denominado consulta cubierta) es una optimización en la que Amazon DocumentDB responde a una consulta en su totalidad a partir de los campos almacenados en un índice, sin leer los documentos subyacentes. Como los valores que necesita una consulta ya están presentes en el índice, Amazon DocumentDB puede omitir la búsqueda de todos los documentos coincidentes, lo que evita el coste aleatorio I/O y de decodificación que supone leer documentos completos. En el plan de consultas, aparece como `IXONLYSCAN` etapa un escaneo que solo incluya índices, en lugar de un escaneo (escaneo de índices) o `IXSCAN` `COLLSCAN` (escaneo de colección). Esto reduce I/O y puede reducir considerablemente la latencia de las consultas compatibles.

Una consulta es una buena opción para un escaneo que solo incluya índices cuando solo toque un número reducido de campos que estén todos en un mismo índice, especialmente cuando los documentos en sí son grandes. Algunos ejemplos comunes son los paneles, los resúmenes de análisis y las búsquedas que leen algunos campos de un índice compuesto más amplio.

Los resultados de una consulta son los mismos independientemente de que se utilice o no un análisis que solo contenga índices. Solo cambian el plan de consulta y la cantidad I/O que realiza.

> **Note**
>
> Index-only la compatibilidad con el escaneo depende de la versión del motor de Amazon DocumentDB:
>
> - `find`Las proyecciones cubiertas que incluyen más de una proyección que no es de `_id` campo requieren la versión 8.0.2 o posterior del motor. En versiones anteriores, una `find` proyección cubierta podía incluir como máximo una proyección que no fuera de `_id` campo. Una proyección con dos o más campos vuelve a ser un escaneo de índice o de colección normal.
> - Index-only los análisis en busca de canales de agregación, como los cubiertos`$group`, requieren la versión 8.0.2 o posterior del motor.

Cuando Amazon DocumentDB utiliza un escaneo que solo incluye índices

Amazon DocumentDB considera la posibilidad de escanear solo índices para las siguientes operaciones cuando un índice único contiene todos los campos que lee la operación.

Proyecciones de búsqueda cubiertas

Una `find` consulta cumple los requisitos cuando su proyección es una simple inclusión de uno o más campos de nivel superior y todos los campos proyectados están presentes en un único índice. Por ejemplo, si el índice compuesto está activado`{ customerId: 1, status: 1, total: 1 }`, se incluye la siguiente consulta. Los tres campos proyectados se encuentran en el índice y `_id` se excluyen explícitamente. (Para cubrir una proyección con más de un campo que no sea un `_id` campo, se requiere la versión 8.0.2 o posterior del motor).

```
db.orders.createIndex({ customerId: 1, status: 1, total: 1 })

db.orders.find(
  { customerId: 42 },
  { _id: 0, customerId: 1, status: 1, total: 1 }
)
```

El plan ganador utiliza un `IXONLYSCAN` escenario y nunca saca del almacén los documentos correspondientes:

```
{
  "stage": "IXONLYSCAN",
  "indexName": "customerId_1_status_1_total_1",
  "direction": "forward",
  "indexCond": { "$and": [ { "customerId": { "$eq": 42 } } ] }
}
```

> **Note**
>
> En una `find` consulta cubierta, los campos del resultado se devuelven en orden de índice (el orden en que aparecen los campos en la definición del índice), no en el orden en que se

enumeraron en la proyección ni en el orden en que se almacenaron en el documento. Esto coincide con el comportamiento de MongoDB en las consultas cubiertas. En cambio, una consulta a la que se responde mediante un escaneo de la colección devuelve los campos en el orden de los documentos almacenados.

Se cubrieron **las agregaciones de $group**

Por lo general, una `$group` etapa lee solo unos pocos campos (la clave del grupo y las entradas de sus acumuladores), pero a menudo se ejecuta sin un encabezado. `$match` Sin un índice que lo cubra, Amazon DocumentDB lee y decodifica todos los documentos completos de la colección solo para incluir esos pocos campos en la agrupación.

Index-only Los análisis para detectar las canalizaciones de agregación, incluidas `$group` las etapas cubiertas, están disponibles en la versión 8.0.2 y posteriores del motor.

Cuando un solo índice contiene todos los campos que lee el prefijo de la canalización, Amazon DocumentDB puede colocar un escaneo solo para índices debajo del grupo y leer esos campos directamente desde el índice. Pensemos en una canalización que agrupa los pedidos por estado y suma sus totales, con un índice compuesto sobre: `{ status: 1, total: 1 }`

```
db.orders.createIndex({ status: 1, total: 1 })

db.orders.aggregate([
  { $group: { _id: "$status", revenue: { $sum: "$total" } } }
])
```

La canalización es de solo lectura `status` y`total`, al estar ambas en el índice, el grupo se alimenta a partir de un escaneo `IXONLYSCAN` en lugar de un registro:

```
// 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)
```

Se admite un `$sort` encabezado `$match``$skip`,, o `$limit` antes del`$group`. Sin embargo, todos los campos a los que hacen referencia estas etapas (por ejemplo, un campo de `$match` filtro o una `$sort` clave) también deben estar presentes en el mismo índice para cubrir la canalización. Si falta algún campo al que se hace referencia en el índice, Amazon DocumentDB recurre a un escaneo de recopilación o índice.

> **Note**
>
> A diferencia de una `find` proyección, a `$group` no incluye el documento de forma implícita. `_id` `_id`En a `$group` es la clave del grupo, no el identificador del documento, por lo que no es necesario que el índice contenga el documento `_id` para cubrir el proceso.

Requisitos para una consulta cubierta

Para que Amazon DocumentDB utilice un escaneo que solo incluya índices, se deben cumplir todos los siguientes requisitos:

- Un único índice contiene todos los campos que lee la consulta. Esto incluye los campos proyectados y la clave de grupo, las entradas del acumulador y cualquier campo de filtro u clasificación para una canalización cubierta`$group`. `find` Si falta un solo campo en el índice, la consulta no se puede cubrir.
- La proyección es una inclusión de campos de nivel superior (para`find`). Las proyecciones de exclusión, los campos calculados y los operadores de proyección, como `$slice` y, no `$elemMatch` son aptos.
- El **`_id`** campo está contabilizado (para`find`). Como `_id` se incluye de forma predeterminada, un índice compuesto común, por ejemplo, `{ a: 1, b: 1 }` no cubre`find({}, { a: 1, b: 1 })`: lo implícito no `_id` está en el índice. Puede excluirlo `_id` de la proyección (`{ _id: 0, a: 1, b: 1 }`) o añadirlo `_id` al índice.
- El índice cumple plenamente con el filtro. Si Amazon DocumentDB no puede convertir parte del predicado de consulta en un enlace a un índice, conserva esa parte como filtro residual, lee el documento para evaluarlo y no convierte el escaneo en un escaneo solo para índices. Los operadores como`$mod`, `$ne``$nin`, y `$exists` dejan un filtro residual.
- Ningún campo proyectado es una columna de varias claves (matriz). Consulte Restricciones.

Restricciones

En los siguientes casos, no se puede usar un escaneo solo de índices y volver a un escaneo de índice o colección normal. Este es el comportamiento esperado y nunca afecta a la exactitud de los resultados.

- Campos multiclave (matriz). Si un campo ha mantenido alguna vez un valor de matriz en algún documento, su columna de índice es de varias claves. Un índice de varias claves almacena una entrada por elemento de la matriz, por lo que la matriz original no se puede reconstruir a partir del índice. No se incluye ninguna consulta que lea un campo de este tipo.
- Rutas punteadas o anidadas. Una proyección o una referencia de campo en una ruta anidada como, por ejemplo, no `"address.city"` es apta, incluso si el índice contiene esa ruta.
- Proyecciones de exclusión, calculadas o de operadores. Las proyecciones como `{ b: 0 }`, `{ total: { $add: ["$price", "$tax"] } }`, o `{ a: { $slice: 5 } }` requieren el documento completo y no están cubiertas.
- Whole-document referencias agregadas. No `$group` se puede incluir una referencia a todo el documento (por ejemplo, `{ $push: "$$ROOT" }`), porque ningún índice puede reconstruir todo el documento.
- La remodelación de las etapas anteriores. **`$group`** Una etapa que cambia la forma de los documentos (como `$project`, `$addFields` `$unwind`, o `$lookup`) que aparecen antes de que `$group` descalifique al oleoducto para que no pueda cubrirse.
- La intercalación no coincide en los predicados de cadenas. Si la intercalación de la consulta difiere de la intercalación del índice, el índice no puede mostrar los predicados de cadena y deja un filtro residual, lo que obliga a buscar un documento. Los predicados numéricos y basados en tipos no se ven afectados.

Confirmación de un análisis que solo incluye índices

Utilice el `explain("executionStats")` resultado para confirmar que la consulta se ha realizado mediante un escaneado que solo contiene índices y que no afecta a los documentos. Busque dos cosas:

- El plan ganador usa el `IXONLYSCAN` escenario.
- El `docsExamined` valor de esa etapa es `0` (y `totalDocsExamined` es `0`), lo que significa que la consulta se respondió en su totalidad desde el índice sin leer ningún 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` valor distinto de cero en un `IXONLYSCAN` escenario significa que el índice contenía los datos, pero Amazon DocumentDB siguió comprobando el almacenamiento para confirmar que cada versión del documento estaba visible. Esto puede ocurrir cuando la colección contiene escrituras recientes cuyas versiones obsoletas del documento no se han recuperado todavía. Una vez que la recolección de basura procesa esos documentos, `docsExamined` pasa a`0`. En ese momento, la consulta se da cuenta de todos los beneficios del escaneo cubierto. Para obtener más información sobre la recolección de basura, consulteRecopilación de elementos no utilizados en Amazon DocumentDB. Para obtener más información acerca de la métrica `docsExamined`, consulte Análisis del plan de consultas.

Prácticas recomendadas

- Diseñe índices compuestos que contengan los campos en los que proyectan o agrupan sus consultas más frecuentes, de modo que esas consultas puedan cubrirse.
- Excluya `_id` de `find` las proyecciones (`{ _id: 0, ... }`) a menos que `_id` forme parte del índice de cobertura.
- Evite indexar campos de matriz (con varias claves) cuando pretenda cubrir una consulta, ya que las columnas con varias claves descalifican la ruta que solo contiene índices.
- Permita que la recolección de basura esté a la altura de su carga de trabajo de escritura, de modo que los escaneos cubiertos eviten el acceso a los documentos y se queden así. `docsExamined 0` Para obtener más información, consulte Recopilación de elementos no utilizados en Amazon DocumentDB.

Planificar filtros de caché

Un filtro de caché del plan, también denominado filtro de índices, restringe el conjunto de índices que el planificador de consultas puede considerar para una forma de consulta específica. Cuando una consulta coincide con una forma que tiene un filtro, el planificador elige un plan solo entre los índices mencionados en ese filtro en lugar de evaluar todos los índices de la colección.

Una forma de consulta es la combinación del predicado de la consulta, la especificación de clasificación y el espacio de nombres de la colección, con los valores del predicado normalizados. Dos consultas que solo difieren en sus valores comparten una forma, por lo que un solo filtro las cubre todas. En la salida de`planCacheListFilters`, los valores normalizados se muestran como`"@"`.

Un filtro de caché del plan le permite fijar el plan para evitar elegir un índice más lento debido a la distribución de los datos o a los cambios en el índice. Los filtros tienen las siguientes propiedades:

- No es necesario cambiar la aplicación. Los filtros se configuran con un comando de base de datos y se aplican en el servidor, por lo que puede mitigar una regresión sin necesidad de implementar código. Esta es la principal diferencia con respecto a a`hint`, que se debe agregar en cada sitio de llamada.
- El ámbito es una forma de consulta. Una forma se define por la colección, el predicado y la clasificación, no por la consulta completa, por lo que un filtro restringe solo las consultas que coinciden con esa forma. Otras consultas de la misma colección se siguen planificando con normalidad.

- El cambio es duradero. Los filtros persisten durante los reinicios de las instancias y los parches del motor. Al eliminar la colección, se eliminan sus filtros.
- El cambio es reversible. Al borrar el filtro, se devuelve la forma a la planificación normal basada en los costos, por lo que un filtro es una forma de bajo riesgo de estabilizar una carga de trabajo mientras se investiga la causa subyacente.

Temas

- Comandos admitidos
- Formas de consulta compatibles
- Configurar, enumerar y borrar filtros
- Comprobar que se ha aplicado un filtro
- Ejemplos
- Comportamiento con sugerencias y cambios en el índice

Comandos admitidos

Los filtros de caché del plan requieren la versión 2.0 o posterior de Planner. El comando al que se puede aplicar un filtro depende de la versión del planificador y de la versión secundaria de Amazon DocumentDB, como se muestra en la tabla siguiente.

Soporte de comandos para los filtros de caché del plan

Comando	Versión mínima del planificador	Versión secundaria mínima
`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

Comando	Versión mínima del planificador	Versión secundaria mínima
`aggregate`	3.0	8.0.2

> **Note**
>
> La versión 3.0 de Planner solo está disponible en Amazon DocumentDB 8.0. Como los `aggregate` comandos `distinct` y requieren la versión 3.0 del planificador, los filtros de caché del plan para esos dos comandos solo están disponibles en Amazon DocumentDB 8.0. En Amazon DocumentDB 5.0, que admite la versión 2.0 del planificador, los filtros se aplican a los `count` comandos `findupdate`,`delete`, y`findAndModify`.

Para el `aggregate` comando, la forma se toma desde la parte frontal de la canalización: la línea inicial `$match` proporciona el filtro e `$sort` inmediatamente después proporciona la clasificación. `$skip`y `$limit` no afectan a la forma, ni tampoco a las etapas que se encuentran más allá de ese punto.

Amazon DocumentDB optimiza la canalización antes de planificarla, y un filtro de caché del plan se compara con la canalización optimizada. Por lo tanto, el filtro y la clasificación que ve el planificador pueden diferir de las etapas que escribió. Por ejemplo, dos `$match` etapas adyacentes en la parte delantera de una canalización se combinan en un solo `$and` predicado:

```
db.orders.aggregate([
  { $match: { status: "SHIPPED" } },
  { $match: { region: "us-east-1" } },
  { $group: { _id: "$customerId", total: { $sum: "$amount" } } }
])
```

El planificador ve uno `$match` activado`{$and: [{status: ...}, {region: ...}]}`, por lo que el filtro debe configurarse en ese predicado combinado.

```
db.runCommand({planCacheSetFilter: "orders",
query: { $and: [ { status: "SHIPPED" }, { region: "us-east-1" } ] },
indexes: [ "status_1_region_1" ]})
```

Un filtro establecido en la colección extranjera de una `$lookup` o varias `$graphLookup` etapas controla el índice utilizado para escanear esa colección extranjera.

Formas de consulta compatibles

A partir de Amazon DocumentDB 5.0.2 y 8.0.2, los siguientes operadores pueden aparecer en forma de consulta con un filtro de caché de planes:

- `$text`. La cadena de búsqueda está normalizada, por lo que las consultas que solo difieren en los términos que buscan comparten una forma.
- `$near` y `$nearSphere`. Ambos operadores se normalizan con la misma forma. Las coordenadas `$minDistance` y no `$maxDistance` forman parte de la forma.
- `$geoWithin` y `$geoIntersects`. El tipo de geometría GeoJSON forma parte de la forma, por lo que una `Polygon` consulta y una `MultiPolygon` consulta son formas diferentes. Las coordenadas no forman parte de la forma.
- `$regex`, cuando se usa directamente en un campo. El patrón y las opciones están normalizados, por lo que todas las expresiones regulares del mismo campo comparten la misma forma.

La configuración de un filtro produce un error y se produce el código de error 303 para una forma de consulta que utilice `$jsonSchema` o`$sampleRate`. `$expr` También se produce un error en las siguientes formas. Una consulta que utilice cualquiera de ellas se planifica sin ningún filtro:

- Un `$regex` anidado dentro de `$in``$nin`, o`$all`. Una matriz que mezcla expresiones regulares con valores literales no se puede evaluar comparándola con un índice, por lo que nunca se podría aplicar un filtro a esa forma.
- Una especie de especificación de`{$natural: 1}`. Una `$natural` clasificación solicita un escaneo de la colección, lo cual es incompatible con restringir el planificador a un índice.

Configurar, enumerar y borrar filtros

Para configurar un filtro de caché del plan, utilice el `planCacheSetFilter` comando. Los `sort` campos `query` y juntos especifican la forma de la consulta y `indexes` muestran los nombres de índice que el planificador puede tener en cuenta para esa forma.

```
db.runCommand({
planCacheSetFilter: <collection>,
```

```
query: <query>,
sort: <sort>, // optional
indexes: [ <index1>, <index2>, ...],
comment: <any> // optional
})
```

Por ejemplo, el siguiente comando restringe el planificador al `a_1` índice de la forma `{a: {$eq: "@"}, b: {$eq: "@"}}` sin ningún tipo de clasificación:

```
db.runCommand({planCacheSetFilter: "foo",
query: { a: 1, b: 1 },
sort: {},
indexes: [ "a_1" ]})
```

Al establecer un filtro para una forma que ya tiene uno, se reemplaza el filtro existente. Tras ejecutar los dos comandos siguientes, el filtro de la forma `{a: {$eq: "@"}, b: {$eq: "@"}}` es`["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" ]})
```

Para mostrar todos los filtros de una colección, utilice el `planCacheListFilters` comando:

```
db.runCommand({planCacheListFilters: <collection>})
```

El resultado muestra cada forma almacenada con sus valores normalizados y su lista de índices permitidos:

```
{
"filters" : [
{
"query" : { "a" : { "$eq" : "@" } },
"sort" : { },
```

```
"indexes" : [ "a_1" ]
},
{
"query" : { "a" : { "$gt" : "@" } },
"sort" : { "a" : 1 },
"indexes" : [ "a_1_b_1" ]
}
],
"ok" : 1
}
```

El operador se conserva y solo se reemplaza el valor por. "@" Una igualdad implícita, tal como, `{a: 1}` aparece en su forma explícita`{"a": {"$eq": "@"}}`, y los elementos de una `$in` matriz se reemplazan como un todo, como`{"a": {"$in": "@"}}`. Un conjunto de filtros sin clasificación aparece junto con un `sort` documento vacío.

Para eliminar los filtros, utilice el `planCacheClearFilters` comando.

```
db.runCommand({
planCacheClearFilters: <collection>,
query: <query pattern>, // optional
sort: <sort specification>, // optional
comment: <any> // optional
})
```

Para borrar todos los filtros de la colección, omite ambos `query` y`sort`:

```
db.runCommand({planCacheClearFilters: "foo"})
```

Suministra ambos `query` y `sort` borra solo el filtro que tiene ese tipo exacto. Los valores introducidos `query` se normalizan, por lo que cualquier valor representativo funciona. Para borrar solo el filtro que se estableció sin ordenar, pase un `sort` documento vacío:

```
db.runCommand({planCacheClearFilters: "foo", query: {a: 1}, sort: {a: 1}})

db.runCommand({planCacheClearFilters: "foo", query: {a: 1}, sort: {}})
```

Comprobar que se ha aplicado un filtro

El `explain` resultado incluye dos campos que indican el estado del filtro de caché del plan:

- `indexFilterSet`es `true` cuando existe un filtro en la colección para la forma de la consulta.
- `indexFilterApplied`es `true` solo cuando la consulta se planificó con un índice de la lista de ese filtro.

A partir de Amazon DocumentDB 5.0.2 y 8.0.2, estos campos se incluyen en todos los comandos que admiten los filtros de caché de planificación y en los resultados del generador de perfiles de esos comandos. En versiones secundarias anteriores, solo se indicaban para el comando. `find` Para obtener más información sobre la lectura y la explicación del resultado, consulteAnálisis del plan de consultas.

Un filtro puede coincidir con la forma de una consulta sin aplicarlo. Si ninguno de los índices de la lista del filtro sirve para la consulta, el planificador vuelve a planificar sin el filtro y elige un índice según el costo. En ese caso `indexFilterSet` es `true` y es. `indexFilterApplied` `false` Esto ocurre cuando los índices con nombre no cubren los campos de la consulta, cuando los índices con nombre no existen o cuando una `$regex` forma no puede generar límites de índice, como un patrón desanclado o que no distingue mayúsculas de minúsculas.

Ejemplos

En los siguientes ejemplos se utiliza una `orders` colección, y una `customers` colección, para el ejemplo, con estos índices: `$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
```

Ejemplo: anclar una **`consulta de`** búsqueda a un índice compuesto

Imagínese una consulta que filtra `status` y ordena según`orderDate`:

```
db.orders.find({ status: "SHIPPED" }).sort({ orderDate: 1 })
```

El índice compuesto `status_1_orderDate_1` satisface tanto el predicado como la clasificación, por lo que no es necesaria una clasificación por separado. Si el planificador lo elige`status_1`, la consulta tiene que ordenar sus resultados, lo que añade una `SORT` etapa y se encarece a medida que aumenta el número de pedidos coincidentes. Para restringir el planificador al índice compuesto de esta forma:

```
db.runCommand({planCacheSetFilter: "orders",
query: { status: "SHIPPED" },
sort: { orderDate: 1 },
indexes: [ "status_1_orderDate_1" ]})
```

Como los valores se normalizan sin tener en cuenta la forma, este filtro también cubre `{ status: "PENDING" }{ status: "CANCELLED" }`, y todos los demás valores de `status` la misma clasificación. No cubre el mismo predicado con un orden diferente o sin ningún tipo de clasificación, que son formas independientes. Confirme que el filtro se ha aplicado con`explain`:

```
db.orders.find({ status: "SHIPPED" }).sort({ orderDate: 1 }).explain()
```

Los informes de salida `indexFilterSet: true` y`indexFilterApplied: true`, y los nombres de las `IXSCAN` etapas`status_1_orderDate_1`.

Ejemplo: fijar una **actualización** a un índice selectivo

Una actualización tiene que localizar los documentos que se van a modificar antes de poder escribirlos, y esa búsqueda utiliza un índice del mismo modo que lo hace una lectura. Cuando más de un índice puede servir de base a un predicado, el índice que elija el planificador determina cuántos documentos examina la actualización.

```
db.orders.updateMany(
  { customerId: 4815, status: "PENDING" },
  { $set: { status: "CANCELLED" } }
)
```

Ambos `customerId_1` y `status_1` pueden servir para este predicado. Para anclar la actualización a`customerId_1`:

```
db.runCommand({planCacheSetFilter: "orders",
query: { customerId: 4815, status: "PENDING" },
indexes: [ "customerId_1" ]})
```

Los filtros de los comandos de escritura requieren Amazon DocumentDB 5.0.2 u 8.0.2 y la versión 2.0 o posterior del planificador. Compruébelo con una `explain` de las actualizaciones:

```
db.runCommand({explain: {update: "orders",
updates: [{ q: { customerId: 4815, status: "PENDING" },
```

```
            u: { $set: { status: "CANCELLED" } },
            multi: true }]}})
```

El plan ganador es UPDATE superar una IXSCAN etapa`customerId_1`. Se aplica el mismo filtro a `delete` los `findAndModify` comandos que utilizan esta forma de consulta, ya que la forma no incluye el tipo de escritura.

Ejemplo: anclar una **`canalización`** agregada y una colección externa de **`$lookup`**

En el caso de una agregación, la forma se toma de la `$match` fase inicial e `$sort` inmediatamente después de ella. Las etapas posteriores, como `$group` o`$project`, no forman parte de la figura.

```
db.orders.aggregate([
  { $match: { status: "SHIPPED" } },
  { $group: { _id: "$customerId", total: { $sum: "$amount" } } }
])
```

Un filtro establecido en el `$match` predicado controla el índice que se utiliza para leer: `orders`

```
db.runCommand({planCacheSetFilter: "orders",
query: { status: "SHIPPED" },
indexes: [ "status_1" ]})
```

Los filtros también se aplican a la colección a la que se une una `$graphLookup` etapa `$lookup` o. El filtro se establece en la colección externa, no en la colección con la que se ejecuta la canalización. En la siguiente canalización, `orders` está la colección externa y `customers` es la colección externa:

```
db.orders.aggregate([
  { $lookup: { from: "customers",
               localField: "customerId",
               foreignField: "customerId",
               as: "customer" } }
])
```

Los sondeos se unen `customers` una vez por documento externo, por lo que el índice elegido allí se aplica repetidamente y su coste se multiplica por todo el proceso. Para fijar ese escaneo a un índice específico, active el filtro `customers` para la forma de igualdad de la clave de unión:

```
db.runCommand({planCacheSetFilter: "customers",
query: { customerId: 1 },
```

```
indexes: [ "customerId_1" ]})
```

Dado que los valores se normalizan fuera de la forma, el valor 1 anterior es un marcador de posición y cualquier valor produce el mismo filtro. Cuando se aplica un filtro de recopilación extranjera, aparecen el nivel superior y los `indexFilterApplied` campos, `indexFilterSet` y el índice en `true` el que se basa el filtro seleccionado aparece en la parte externa del plan ganador. Los filtros activados `aggregate` requieren Amazon DocumentDB 8.0.2 y Planner versión 3.0.

Comportamiento con sugerencias y cambios en el índice

- Un filtro de caché del plan tiene prioridad sobre un`hint`. Amazon DocumentDB aplica primero el filtro y, a continuación, aplica la sugerencia dentro de la lista de índices permitidos del filtro. Si la sugerencia nombra un índice permitido por el filtro, se utiliza ese índice. Si la sugerencia nombra un índice que el filtro no permite, la sugerencia se ignora y el planificador elige entre los índices permitidos en función del costo. Ambas formas de sugerencia se comportan de la misma manera, tanto si se asigna un nombre al índice como si se indica su patrón clave.

  Por ejemplo, si la colección `foo` contiene los índices`a_1`, y `b_1a_1_b_1`, y se establece un filtro en la forma `{query: {a: {$eq: "@"}, b: {$eq: "@"}}, sort: {a: 1}}` con la lista de índices`["a_1", "a_1_b_1"]`, se `db.foo.find({ a: 10, b: 20 }).sort({a: 1}).hint({ a: 1 })` utiliza la ejecución`a_1`, ya que aparece en la sugerencia y el filtro lo permite.
- Eliminar un índice no modifica los filtros. El filtro conserva el nombre del índice. Mientras falta ese índice, la planificación lo omite. Si otro índice de la lista del filtro puede servir para la consulta, el filtro seguirá aplicándose. El `indexFilterApplied` informe `false` solo se publica cuando ninguno de los índices con nombre está disponible. Al volver a crear un índice con el mismo nombre, vuelve a ser elegible.
- Al eliminar una colección, se eliminan todos los filtros de esa colección.

Prácticas recomendadas para Amazon DocumentDB

Conozca las prácticas recomendadas para trabajar con Amazon DocumentDB (compatible con MongoDB). Esta sección se actualiza continuamente a medida que se identifican nuevas prácticas recomendadas.

Temas

- Directrices operativas básicas

- Determinación del tamaño de las instancias
- Uso de índices
- Prácticas recomendadas de seguridad
- Optimización de costos
- Uso de métricas para identificar los problemas de rendimiento
- TTL y cargas de trabajo de series temporales
- Migraciones
- Uso de grupos de parámetros de clúster
- Consultas de canalización de agregación
- BatchInsert y BatchUpdate

Directrices operativas básicas

A continuación se detallan las directrices operativas básicas que se deben seguir al trabajar con Amazon DocumentDB. El acuerdo de nivel de servicios de Amazon DocumentDB requiere que se sigan estas directrices.

- Implemente un clúster formado por dos o más instancias de Amazon DocumentDB en dos zonas de disponibilidad de AWS . Para las cargas de trabajo de producción, implemente un clúster compuesto por tres o más instancias de Amazon DocumentDB en tres zonas de disponibilidad.
- Utilice el servicio dentro de los límites de servicio indicados. Para obtener más información, consulte Cuotas de Amazon DocumentDB.
- Monitorice el uso de la memoria, la CPU, las conexiones y el almacenamiento. Para ayudarlo a mantener el rendimiento y la disponibilidad del sistema, configure Amazon CloudWatch para que le notifique cuando cambien los patrones de uso o cuando se acerque la capacidad de su implementación.
- Escale las instancias verticalmente cuando se esté acercando a los límites de la capacidad. Las instancias deben aprovisionarse con suficientes recursos informáticos (es decir, RAM, CPU, etc.) para adaptarse a aumentos imprevistos en la demanda de las aplicaciones.
- Configure el periodo de retención de copia de seguridad para ajustarlo a su objetivo de punto de recuperación.
- Pruebe la conmutación por error del clúster para comprender cuánto tiempo tarda el proceso en su caso de uso. Para obtener más información, consulte Conmutación por error de Amazon DocumentDB.

- Conéctese al clúster de Amazon DocumentDB utilizando el punto de conexión del clúster (consulte Puntos de conexión de Amazon DocumentDB) y el modo de conjunto de réplicas (consulte Conexión a Amazon DocumentDB como conjunto de réplicas) para minimizar el impacto de una conmutación por error en la aplicación.
- Elija una configuración de preferencia de lectura de controlador que maximice el escalado de lectura mientras cumple con los requisitos de coherencia de lectura de su aplicación. Las preferencias de lectura `secondaryPreferred` permiten las lecturas de réplica y libera la instancia principal para hacer más trabajo. Para obtener más información, consulte Opciones de preferencia de lectura.
- Diseñe su aplicación para que sea resistente en el caso de que se produzcan errores de la red y de la base de datos. Utilice el mecanismo de errores del controlador para distinguir entre errores temporales y errores persistentes. Reintente los errores temporales mediante un mecanismo de retroceso exponencial cuando sea apropiado. Asegúrese de que la aplicación tenga en cuenta la coherencia de datos al implementar la lógica de reintentos.
- Habilite la protección contra eliminación de clústeres para todos los clústeres de producción o cualquier clúster que tenga datos importantes. Antes de eliminar un clúster de Amazon DocumentDB, tome una instantánea final. Si vas a implementar recursos con CloudFormation, habilita la protección por terminación. Para obtener más información, consulte Protección de terminación y protección de eliminación.
- Cuando crea un clúster de Amazon DocumentDB, la `--engine-version` es un parámetro opcional que es el valor predeterminado de la última versión principal del motor. La versión predeterminada actual del motor es la 5.0.0 (nota: Amazon DocumentDB 8.0 está disponible, pero debe especificarse de forma explícita). Cuando se publiquen nuevas versiones principales del motor, la versión del motor predeterminada para `--engine-version` se actualizará para reflejar la última versión principal del motor. Como resultado, para las cargas de trabajo de producción, y especialmente las que dependen de la creación de scripts, la automatización o CloudFormation las plantillas, especifique explícitamente la versión principal `--engine-version` deseada.

Determinación del tamaño de las instancias

Uno de los aspectos más importantes a la hora de elegir un tamaño de instancia en Amazon DocumentDB es la cantidad de RAM de la memoria caché. Amazon DocumentDB reserva un tercio de la RAM para sus propios servicios, lo que significa que solo dos tercios de la RAM de la instancia están disponibles para la memoria caché. Es una práctica recomendada de rendimiento de Amazon DocumentDB elegir un tipo de instancia con suficiente RAM para que su conjunto de trabajo (es

decir, datos e índices) quepa en la memoria. Disponer de instancias del tamaño adecuado ayudará a optimizar el rendimiento general y, potencialmente, a minimizar los I/O costos. Puede utilizar la calculadora de tamaño de Amazon DocumentDB de terceros para estimar el tamaño de la instancia para una carga de trabajo concreta.

Para determinar si el conjunto de trabajo de su aplicación cabe en la memoria, supervise el `BufferCacheHitRatio` uso CloudWatch de Amazon en cada instancia de un clúster que esté bajo carga.

La `BufferCacheHitRatio` CloudWatch métrica mide el porcentaje de datos e índices enviados desde la memoria caché de una instancia (en comparación con el volumen de almacenamiento). En términos generales, el valor de `BufferCacheHitRatio` debe ser lo más alto posible, ya que la lectura de datos de la memoria del conjunto de trabajo es más rápida y rentable que la lectura del volumen de almacenamiento. Si bien es deseable mantener `BufferCacheHitRatio` lo más cerca posible del 100 %, el mejor valor alcanzable dependerá de los patrones de acceso y los requisitos de rendimiento de su aplicación. Para mantener el `BufferCacheHitRatio` más alto posible, se recomienda que las instancias del clúster cuenten con suficiente RAM para poder ajustar los índices y el conjunto de datos de trabajo en la memoria.

Si sus índices no caben en la memoria, verá un valor de `BufferCacheHitRatio` inferior. La lectura continua desde el disco implica I/O costos adicionales y no es mejor que leer desde la memoria. Si la proporción de `BufferCacheHitRatio` es inferior a la esperada, escale verticalmente el tamaño de la instancia del clúster para proporcionar más RAM, de forma que los datos del conjunto de trabajo quepan en la memoria. Si escalar la clase de instancia provoca un aumento drástico de `BufferCacheHitRatio`, entonces el conjunto de trabajo de su aplicación no cabe en la memoria. Continúe con el escalado vertical hasta que `BufferCacheHitRatio` ya no aumente drásticamente después de una operación de escalado. Para obtener información acerca del monitoreo de las métricas de una instancia, consulte Métricas de Amazon DocumentDB.

Dependiendo de la carga de trabajo y los requisitos de latencia, puede ser aceptable que la aplicación tenga valores más altos de `BufferCacheHitRatio` durante el uso de estado constante, pero tenga una caída periódica de `BufferCacheHitRatio` a medida que las consultas analíticas que necesitan analizar una colección completa se ejecutan en una instancia. Estas caídas periódicas en `BufferCacheHitRatio` pueden manifestarse como una latencia más alta para consultas posteriores que necesitan volver a llenar los datos del conjunto de trabajo desde el volumen de almacenamiento de nuevo en la caché del búfer. Pruebe sus cargas de trabajo en un entorno de preproducción con una carga de trabajo de producción representativa primero para comprender las

características de rendimiento y **`BufferCacheHitRatio`** antes de implementar la carga de trabajo en producción.

`BufferCacheHitRatio` es una métrica específica de instancia, por lo que las distintas instancias dentro del mismo clúster pueden tener valores de `BufferCacheHitRatio` diferentes, dependiendo de cómo se distribuyen las lecturas entre la instancia principal y la de réplica. Si la carga de trabajo operativa no puede controlar los aumentos periódicos de latencia por volver a llenar la caché del conjunto de trabajo después de ejecutar consultas analíticas, debe intentar aislar la caché del búfer de la carga de trabajo normal de la de las consultas analíticas. Puede lograr un aislamiento completo de `BufferCacheHitRatio` dirigiendo las consultas operativas a la instancia principal y las consultas analíticas solo a las instancias de réplica. También puede lograr un aislamiento parcial dirigiendo consultas analíticas a una instancia de réplica específica, entendiendo que un porcentaje de consultas regulares también se ejecutarán en esa réplica y podrían verse afectadas.

Los valores adecuados de `BufferCacheHitRatio` dependen del caso de uso y los requisitos de la aplicación. No hay un valor máximo o mínimo para esta métrica; solo usted puede decidir si la compensación de un valor de `BufferCacheHitRatio` temporalmente inferior es aceptable desde una perspectiva de costo y rendimiento.

Uso de índices

Creación de índices

Al importar datos en Amazon DocumentDB, debe crear los índices antes de importar grandes conjuntos de datos. Puede utilizar la herramienta de índices de Amazon DocumentDB para extraer índices de una instancia de MongoDB en ejecución o de un directorio `mongodump`, y crear dichos índices en un clúster de Amazon DocumentDB. Para obtener más información acerca de las migraciones, consulte Migración y actualización de Amazon DocumentDB.

Selectividad de índice

Limite la creación de índices a los campos en los que el número de valores duplicados sea inferior al 1% del número total de documentos de la colección. Por ejemplo, si la colección cuenta con 100 000 documentos, solo cree índices en campos donde el mismo valor se produzca 1000 veces o menos.

Elegir un índice con un alto número de valores únicos (es decir, una alta cardinalidad) garantiza que las operaciones de filtro devuelvan un pequeño número de documentos, con lo que se obtiene un buen rendimiento durante los análisis de índices. Un ejemplo de un índice de alta cardinalidad es un

índice único, que garantiza que los predicados de igualdad devuelvan como máximo un documento único. Entre los ejemplos de baja cardinalidad se incluyen un índice sobre un campo booleano y un índice sobre el día de la semana. Debido a su bajo rendimiento, es poco probable que el optimizador de consultas de la base de datos elija índices de cardinalidad baja. Al mismo tiempo, los índices de cardinalidad bajos siguen consumiendo recursos como espacio en disco y. I/Os Como regla general, debe aplicar los índices a campos donde la frecuencia del valor típica sea un 1 % del tamaño total de la colección o menos.

Además, se recomienda crear índices únicamente en campos que se utilizan comúnmente como filtro y buscar regularmente índices no utilizados. Para obtener más información, consulte ¿Cómo analizo el uso de los índices e identifico los índices no utilizados?.

Impacto de los índices en los datos de escritura

Aunque los índices pueden mejorar el rendimiento de las consultas al evitar la necesidad de examinar todos los documentos de una colección, esta mejora tiene algún inconveniente. Para cada índice de una colección, cada vez que se inserta, actualiza o elimina un documento, la base de datos debe actualizar la colección y escribir los campos en cada uno de los índices de la colección. Por ejemplo, si una colección tiene nueve índices, la base de datos debe realizar diez escrituras antes de validar la operación para el cliente. Por lo tanto, cada índice adicional genera latencias de escritura adicionales y un aumento en el almacenamiento utilizado en general. I/O

Las instancias del clúster deben tener el tamaño adecuado para mantener toda la memoria del conjunto de trabajo. Esto evita la necesidad de leer continuamente las páginas de índice del volumen de almacenamiento, lo que repercute negativamente en el rendimiento y genera I/O costos más altos. Para obtener más información, consulte Determinación del tamaño de las instancias.

Para obtener un mejor rendimiento, reduzca el número de índices de sus colecciones, agregando solo los índices necesarios para mejorar el rendimiento de las consultas comunes. Aunque las cargas de trabajo varían, una buena directriz es mantener el número de índices por colección en cinco o menos.

Identificación de índices que faltan

Como práctica recomendada, identifique los índices que faltan de forma regular. Para obtener más información, consulte ¿Cómo identifico los índices que faltan?.

Identificación de índices no utilizados

Como práctica recomendada, identifique y elimine los índices no utilizados de forma regular. Para obtener más información, consulte ¿Cómo analizo el uso de los índices e identifico los índices no utilizados?.

Prácticas recomendadas de seguridad

Para conocer las prácticas recomendadas de seguridad para Amazon DocumentDB, consulte. Prácticas recomendadas de seguridad para Amazon DocumentDB

Optimización de costos

Las siguientes prácticas recomendadas pueden ayudarlo a administrar y minimizar sus costos cuando utilice Amazon DocumentDB. Para obtener información sobre precios, consulte los precios de Amazon DocumentDB (compatible con MongoDB) y las preguntas frecuentes sobre Amazon DocumentDB (compatible con MongoDB).

- Cree alertas de facturación en umbrales del 50 y el 75 % de su factura prevista para el mes. Para obtener más información sobre la creación de alertas de facturación, consulte Creación de una alarma de facturación.
- La arquitectura de Amazon DocumentDB separa el almacenamiento y la computación, por lo que incluso un clúster de una sola instancia es muy duradero. El volumen de almacenamiento del clúster replica los datos de seis formas en tres zonas de disponibilidad, lo que proporciona una durabilidad extremadamente alta con independencia del número de instancias del clúster. Un clúster típico de producción tiene tres o más instancias para proporcionar alta disponibilidad. Sin embargo, puede optimizar los costes mediante un clúster de desarrollo de una única instancia cuando no se requiera alta disponibilidad.
- En las situaciones de desarrollo y pruebas, detenga un clúster cuando ya no sea necesario e inícielo cuando se reanude el desarrollo. Para obtener más información, consulte Detención e inicio de un clúster de bases de datos de Amazon DocumentDB.
- Tanto el TTL como los flujos de cambios se producen I/O cuando se escriben, leen y eliminan los datos. Si ha habilitado estas características pero no las está utilizando en la aplicación, su desactivación puede ayudar a reducir los costos.

Uso de métricas para identificar los problemas de rendimiento

Temas

- Visualización de métricas de rendimiento
- Configurar una CloudWatch alarma
- Evaluación de las métricas de rendimiento
- Evaluación del uso de las instancias de Amazon DocumentDB con métricas CloudWatch
- Ajuste de consultas

Para identificar los problemas de rendimiento causados por la falta de recursos y otros cuellos de botella frecuentes, puede monitorizar las métricas disponibles para su clúster de base de datos de Amazon DocumentDB.

Visualización de métricas de rendimiento

Monitorice las métricas de desempeño con frecuencia para ver los valores medios, máximos y mínimos de diversos intervalos de tiempo. Esto lo ayuda a identificar cuándo se degrada el rendimiento. También puedes configurar CloudWatch las alarmas de Amazon para determinados umbrales métricos, de modo que te avisen si se alcanzan.

Para solucionar los problemas de rendimiento, es importante conocer el desempeño de referencia del sistema. Después de configurar un nuevo clúster y ponerlo en marcha con una carga de trabajo típica, capture los valores promedio, máximo y mínimo de todas las métricas de rendimiento en diferentes intervalos (por ejemplo, 1 hora, 24 horas, 1 semana, 2 semanas). Esto permite hacerse una idea de lo que es normal. Ayuda a obtener comparaciones para las horas con picos y valles de funcionamiento. Puede usar esta información para saber cuándo cae el desempeño por debajo de los niveles estándar.

Puede ver las métricas de rendimiento mediante el Consola de administración de AWS comando o. AWS CLI Para obtener más información, consulte Visualización CloudWatch de datos.

Configurar una CloudWatch alarma

Para configurar una CloudWatch alarma, consulta el artículo Uso de CloudWatch las alarmas de Amazon en la Guía del CloudWatch usuario de Amazon.

Evaluación de las métricas de rendimiento

Una instancia tiene varias categorías diferentes de métricas. La forma de determinar los valores aceptables depende de la métrica.

CPU

- Utilización de la CPU: el porcentaje de la capacidad de procesamiento del equipo que está en uso.

Memoria

- Memoria que se puede liberar: cuánta RAM está disponible en la instancia.
- Uso del espacio de intercambio: cuánto espacio de intercambio usa la instancia en megabytes.

Input/output operaciones

- IOPS de lectura, IOPS de escritura: número medio de operaciones de lectura o escritura en disco por segundo.
- Latencia de lectura, latencia de escritura: tiempo medio de una operación de lectura o escritura en milisegundos.
- Rendimiento de lectura, rendimiento de escritura: número medio de megabytes leídos o escritos en el disco por segundo.
- Profundidad de cola de discos: el número de I/O operaciones que están esperando a escribirse en el disco o leerse desde él.

Tráfico de red

- Rendimiento de recepción de la red, rendimiento de transmisión de la red: velocidad del tráfico de red de entrada y salida de la instancia de base de datos en megabytes por segundo.

Conexiones a base de datos

- Conexiones a base de datos: número de sesiones cliente que están conectadas a la instancia de base de datos.

En general, los valores aceptables para las métricas de desempeño dependen del aspecto de la referencia y de lo que hace la aplicación. Investigue las variaciones coherentes o de las tendencias con respecto a la referencia.

A continuación, se ofrecen algunas recomendaciones y sugerencias sobre tipos concretos de métricas:

- Consumo elevado de CPU: unos valores elevados de consumo de CPU pueden ser adecuados si se ajustan a los objetivos de su aplicación (de rendimiento o simultaneidad, por ejemplo) y son los esperados. Si el consumo de CPU es sistemáticamente superior al 80 %, valore la posibilidad de escalar las instancias.
- Consumo elevado de RAM: si la métrica `FreeableMemory` se sitúa con frecuencia por debajo del 10% de la memoria total de la instancia, valore la posibilidad de escalar las instancias. Para obtener más información sobre lo que ocurre cuando su instancia de DocumentDB sufre una gran presión de memoria, consulte Amazon DocumentDB Resource Governance.
- Uso de intercambio: esta métrica debe mantenerse en cero o en un valor próximo a cero. Si el uso que hace del intercambio es significativo, valore la posibilidad de escalar las instancias.
- Tráfico de red: para el tráfico de red, hable con el administrador de su sistema para saber cuál es el rendimiento esperado para la red de su dominio y para su conexión a Internet. Investigue el tráfico de red si el rendimiento es por sistema inferior al esperado.
- Conexiones a bases de datos: valore la posibilidad de restringir las conexiones a las bases de datos si ve que hay un alto número de conexiones de usuarios junto con una reducción en el rendimiento y el tiempo de respuesta de la instancia. El mejor número de conexiones de usuarios para su instancia varía en función de la clase de instancia y de la complejidad de las operaciones que se estén llevando a cabo. Si hay problemas con cualquier métrica de desempeño, una de las primeras cosas que debe hacer para mejorar el desempeño es ajustar las consultas más frecuentes y más caras para ver si eso reduce la presión existente en los recursos del sistema.

Si sus consultas están ajustadas y el problema persiste, considere la posibilidad de actualizar su clase de instancia de Amazon DocumentDB a una que tenga más recursos (CPU, RAM, espacio en disco, ancho de banda de red, I/O capacidad) relacionados con el problema que está experimentando.

Evaluación del uso de las instancias de Amazon DocumentDB con métricas CloudWatch

Puede utilizar CloudWatch las métricas para observar el rendimiento de las instancias y descubrir si la clase de instancia proporciona recursos suficientes para las aplicaciones. Para obtener información sobre las cuotas de las clases de instancias, consulta Cuotas de instancias y localiza las especificaciones de tu clase de instancia para conocer el rendimiento de tu red.

Si el uso de la instancia está cerca del límite de las clases de instancia, es posible que el rendimiento comience a disminuir. Las CloudWatch métricas pueden confirmar esta situación para que puedas planificar la escalabilidad manual a una clase de instancia más grande.

Combina los siguientes valores de CloudWatch métricas para averiguar si te estás acercando al límite de clases de instancias:

- NetworkThroughput—La cantidad de rendimiento de red que reciben y transmiten los clientes para cada instancia del clúster de Amazon DocumentDB. Este valor de rendimiento no incluye el tráfico de red entre las instancias del clúster y el volumen de almacenamiento del clúster.
- StorageNetworkThroughput—La cantidad de rendimiento de red que cada instancia del clúster de Amazon DocumentDB recibe y envía al volumen de almacenamiento del clúster de Amazon DocumentDB.

Agregue el NetworkThroughput al StorageNetworkThroughput para encontrar el rendimiento de red recibido y enviado al volumen de almacenamiento del clúster de Amazon DocumentDB por cada instancia de su clúster de Amazon DocumentDB. El límite de clases de instancia para su instancia debe ser mayor que la suma de estas dos métricas combinadas.

Puede utilizar las siguientes métricas para revisar detalles adicionales del tráfico de red de las aplicaciones cliente al enviar y recibir:

- NetworkReceiveThroughput—La cantidad de rendimiento de red que recibe de los clientes cada instancia del clúster de Amazon DocumentDB. Este rendimiento no incluye el tráfico de red entre las instancias del clúster y el volumen de almacenamiento del clúster.
- NetworkTransmitThroughput—La cantidad de rendimiento de red que envía a los clientes cada instancia del clúster de Amazon DocumentDB. Este rendimiento no incluye el tráfico de red entre las instancias del clúster y el volumen de almacenamiento del clúster.
- StorageNetworkReceiveThroughput—La cantidad de rendimiento de red que recibe del volumen de almacenamiento del clúster de Amazon DocumentDB cada instancia del clúster.

- StorageNetworkTransmitThroughput—La cantidad de rendimiento de red que cada instancia del clúster envía al volumen de almacenamiento del clúster de Amazon DocumentDB.

Sume todas estas métricas para evaluar cómo está el uso de la red con respecto al límite de las clases de instancias. El límite de las clases de instancias debe ser mayor que la suma de estas métricas combinadas.

Los límites de la red y el uso de la CPU por una instancia son mutuos. Cuando aumenta el rendimiento de la red, también aumenta la utilización de la CPU. La supervisión del uso de la CPU y la red proporciona información sobre cómo y por qué se agotan los recursos.

Para ayudar a minimizar el uso de la red, puede considerar lo siguiente:

- Utilizar una clase de instancia mayor.
- Dividir las solicitudes de escritura en lotes para reducir las transacciones generales.
- Redirigir la carga de trabajo de solo lectura a una instancia de solo lectura.
- Eliminar los índices no utilizados.

Ajuste de consultas

Una de las formas más eficaces de mejorar el rendimiento de clúster es ajustar las consultas más utilizadas y que más recursos consumen para que ejecutarlas resulte más económico.

Puede utilizar el generador de perfiles (consulte Creación de perfiles de operaciones en Amazon DocumentDB) para registrar el tiempo de ejecución y los detalles de las operaciones realizadas en el clúster. El generador de perfiles es útil para monitorizar las operaciones más lentas del clúster para ayudarle a mejorar el rendimiento de las consultas individuales y el rendimiento general del clúster.

También puede utilizar el comando `explain` para descubrir cómo analizar un plan de consulta para una determinada consulta. Utilice esta información para modificar una consulta o una colección subyacente con el fin de mejorar el rendimiento de las consultas (por ejemplo, cómo añadir un índice).

TTL y cargas de trabajo de series temporales

La eliminación de documentos resultante del vencimiento del índice TTL es un proceso muy laborioso. No se garantiza la eliminación de documentos dentro de un periodo específico. Factores como el tamaño de la instancia, la utilización de recursos de la instancia, el tamaño del documento,

el rendimiento general, el número de índices y si los índices y el conjunto de trabajo caben en la memoria pueden afectar al momento en el que el proceso TTL elimina los documentos caducados.

Cuando el monitor TTL elimina sus documentos, cada eliminación genera I/O costos, lo que aumenta su factura. Si las tasas de procesamiento y de borrado por TTL aumentan, deberías esperar una factura más alta debido al aumento del uso. I/O Sin embargo, si no crea un índice TTL para eliminar documentos, sino que los segmenta en colecciones en función del tiempo y simplemente elimina esas colecciones cuando ya no las necesita, no incurrirá en ningún costo de E/S. Esto puede resultar considerablemente más rentable que utilizar un índice TTL.

En el caso de las cargas de trabajo de series temporales, puedes considerar la posibilidad de crear recopilaciones continuas en lugar de un índice TTL, ya que las recopilaciones sucesivas pueden ser una forma mejor de eliminar datos y ser menos intensivas. I/O Si tiene colecciones grandes (especialmente colecciones de más de 1 TB) o I/O los costos de eliminación mediante TTL son un problema, divida los documentos en colecciones en función del tiempo y descarte las colecciones cuando ya no las necesite. Puede crear una colección por día o una por semana, dependiendo de su tasa de incorporación de datos. Aunque los requisitos varían en función de su aplicación, una buena regla general es tener colecciones más pequeñas en lugar de unas pocas colecciones grandes. Eliminar estas colecciones no implica I/O costos y puede ser más rápido y rentable que usar un índice TTL.

Migraciones

Como práctica recomendada para migrar datos a Amazon DocumentDB, cree primero los índices en Amazon DocumentDB y, a continuación, migre los datos. Crear los índices en primer lugar puede reducir el tiempo general y aumentar la velocidad de la migración. Para hacer esto, puede usar la Herramienta de índice de Amazon DocumentDB. Para obtener más información sobre las migraciones, consulte. Migración y actualización de Amazon DocumentDB

También le recomendamos que antes de migrar la base de datos de producción, se recomienda probar completamente la aplicación en Amazon DocumentDB, teniendo en cuenta la funcionalidad, el rendimiento, las operaciones y el costo.

Uso de grupos de parámetros de clúster

Pruebe los cambios del grupo de parámetros del clúster en un clúster de prueba antes de aplicar los cambios a los clústeres de producción. Para obtener información acerca del procedimiento para realizar la copia de seguridad del clúster, consulte Copia de seguridad y restauración en Amazon DocumentDB.

Consultas de canalización de agregación

Si crea una consulta de canalización de agregación con varias etapas y evalúa un solo subconjunto de los datos de la consulta, utilice la etapa `$match` como primera etapa o al principio de la canalización. Usar primero `$match` reducirá el número de documentos que las etapas posteriores tendrán que procesar dentro de la consulta de canalización de agregación, mejorando así el rendimiento de su consulta.

BatchInsert y BatchUpdate

Si realizas una tasa alta de `batchInsert` and/or `batchUpdate` operaciones simultáneas y la cantidad `FreeableMemory` (CloudWatch métrica) pasa a cero en la instancia principal, puedes reducir la simultaneidad de la carga de trabajo de inserción o actualización por lotes o, si no puedes reducir la simultaneidad de la carga de trabajo, aumentar el tamaño de la instancia para aumentar la cantidad de. `FreeableMemory`

Administración de índices de Amazon DocumentDB

Temas

- Creación de índices de Amazon DocumentDB
- Mantenimiento de índices de Amazon DocumentDB

Creación de índices de Amazon DocumentDB

La compilación de índices en Amazon DocumentDB requiere tomar una serie de decisiones:

- ¿Con qué rapidez debe completarse?
- ¿Es posible que no se pueda acceder a la colección mientras se realiza la compilación?
- ¿Qué cantidad de la potencia de procesamiento de una instancia se puede asignar a la compilación?
- ¿Qué tipo de índice se debe crear?

Esta sección le ayuda a responder a estas preguntas y proporciona los comandos y ejemplos de supervisión para crear un índice de Amazon DocumentDB en su colección de clústeres basada en instancias.

Directrices

Las siguientes pautas incluyen los límites básicos y las desventajas de configuración a la hora de crear nuevos índices:

- Compatibilidad con las versiones de Amazon DocumentDB: si bien la indexación de un solo trabajador se admite en todas las versiones de Amazon DocumentDB, la indexación de varios trabajadores solo se admite en las versiones 4.0, 5.0 y 8.0 de Amazon DocumentDB.
- Compensación de rendimiento: aumentar el número de procesos de trabajo en el proceso de creación del índice aumenta la utilización de la CPU y la E/S de lectura en la instancia principal de la base de datos Amazon DocumentDB. Los recursos necesarios para crear un índice nuevo no estarán disponibles para su carga de trabajo en ejecución.
- Clústeres elásticos: la indexación em paralelo no se admite en los clústeres elásticos de Amazon DocumentDB.
- Clústeres sin servidor: la indexación paralela en clústeres sin servidor requiere una capacidad mínima de al menos 32 DCU. Por debajo de este umbral, se acepta el `workers` parámetro, pero la creación del índice se ejecuta con un solo trabajador. La cantidad máxima de trabajadores en paralelo aumenta en función de la capacidad mínima configurada: hay aproximadamente `min_DCU / 8` trabajadores disponibles. Por ejemplo, con 32 DCU puede usar hasta 4 trabajadores y con 64 DCU puede usar hasta 8 trabajadores.

  > **Note**
  >
  > Es necesario reiniciar el clúster después de cambiar la configuración de capacidad mínima para que entre en vigor el nuevo límite de trabajadores en paralelo.

- Número máximo de trabajadores (basado en instancias): el número máximo de trabajadores que puede configurar depende del tamaño de la instancia principal en el clúster de base de datos. Es la mitad del número total de vCPU de la instancia principal del clúster de base de datos. Por ejemplo, puede ejecutar un máximo de 32 procesos de trabajo en una instancia db.r6g.16xlarge que tenga 64 vCPU.

  > **Note**
  >
  > Los trabajadores en paralelo solo se admiten en instancias 2 veces más grandes o más.

- Número mínimo de procesos de trabajo: el número mínimo de procesos de trabajo que puede configurar es uno. La configuración predeterminada para la creación de índices en clústeres basados en instancias es de dos procesos de trabajo. Sin embargo, puede reducir el número de procesos de trabajo a uno mediante la opción “subprocesos de trabajo”. Esto ejecutará el proceso con un solo proceso de trabajo.
- Compresión de índices: Amazon DocumentDB no admite la compresión de índices. Los tamaños de datos para índices pueden ser superiores que cuando utiliza otras opciones.
- Indexación de varias colecciones: la mitad de las vCPU de la instancia principal del clúster de base de datos se pueden usar para los procesos de trabajo configurados que realizan la creación de índices en varias colecciones.
- Tipos de índice: para obtener una explicación completa de los tipos de índice admitidos en Amazon DocumentDB, consulte Cómo indexar en Amazon DocumentDB en el AWS blog de bases de datos.

Introducción

Para iniciar la creación de índices en una colección, utilice el comando `createIndexes`. De forma predeterminada, el comando ejecutará dos elementos de trabajo paralelos, lo que aumentará dos veces la velocidad del proceso de creación del índice.

Por ejemplo, el siguiente proceso de comando muestra cómo crear un índice para el campo “user_name” de un documento y cómo aumentar la velocidad del proceso de indexación a cuatro elementos de trabajo:

1. Cree índices utilizando dos elementos de trabajo paralelos en el clúster:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx"}]})
```

2. Para optimizar la velocidad del proceso de creación del índice, puede especificar el número de procesos de trabajo mediante la opción “subprocesos de trabajo” (`"workers":<number>`) del comando `db.runCommand createIndexes`.

   Aumente la velocidad del proceso a cuatro procesos de trabajo paralelos:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "workers":4}]})
```

> **Note**
>
> Cuanto mayor sea el número de procesos de trabajo, más rápido avanzará la creación del índice. Sin embargo, cuanto mayor sea el número de procesos de trabajo, mayor será el aumento de la carga en las vCPU y la E/S de lectura de la instancia principal. Asegúrese de que el clúster esté lo suficientemente aprovisionado para soportar el aumento de la carga sin degradar otras cargas de trabajo.

Estado del progreso de indexación

El proceso de creación de índices funciona inicializando, escaneando las colecciones, clasificando las claves y, finalmente, insertándolas mediante un generador de índices. El proceso tiene hasta seis etapas cuando se ejecuta en primer plano y hasta nueve etapas cuando se ejecuta en segundo plano. Puede ver las métricas de estado, como el porcentaje de finalización, el número total de bloques de almacenamiento escaneados, las claves clasificadas y las claves insertadas, etapa por etapa.

Supervise el progreso del proceso de indexación mediante el comando `db.currentOp()` del intérprete de comandos de mongo. Una finalización del 100 % de la última etapa indica que todos los índices se han creado correctamente:

```
db.currentOp({"command.createIndexes": { $exists : true } })
```

> **Note**
>
> La visualización del estado del progreso de la indexación se admite solamente en Amazon DocumentDB 5.0.

Tipos de creación de índices

Los cuatro tipos de compilaciones de índices son:

- Primer plano: la compilación del índice en primer plano bloquea todas las demás operaciones de la base de datos hasta que se crea el índice. La compilación en primer plano de Amazon DocumentDB consta de cinco etapas.

- Primer plano (único): las compilaciones de índices en primer plano de un solo documento (únicas) bloquean otras operaciones de la base de datos como las compilaciones en primer plano normales. A diferencia de la compilación básica en primer plano, la compilación única utiliza una etapa adicional (clasificación de claves 2) para buscar claves duplicadas. La compilación en primer plano (única) consta de seis etapas.
- Fondo: la compilación del índice en segundo plano permite que otras operaciones de la base de datos se ejecuten en primer plano mientras se crea el índice. La compilación de fondo de Amazon DocumentDB consta de ocho etapas.
- Fondo (único): las compilaciones de índices en segundo plano de un solo documento (único) permiten que otras operaciones de la base de datos se ejecuten en primer plano mientras se crea el índice. A diferencia de la compilación básica de fondo, la compilación única utiliza una etapa adicional (clasificación de claves 2) para buscar claves duplicadas. La compilación de fondo (única) se compone de nueve etapas.

Etapas de compilación del índice

Etapa	Primer plano	Primer plano (único)	Introducción	Fondo (único)
Inicializando	1	1	1	1
creando índice: inicializando	2	2	2	2.
creando índice: escaneando colección	3	3	3	3
creando índice: clasificando claves 1	4	4	4	4
creando índice: clasificando claves 2		5		5

Etapa	Primer plano	Primer plano (único)	Introducción	Fondo (único)
creando índice: insertando claves	5	6	5	6
validando: escaneando índice			6	7
validando: clasificando tuplas			7	8
validando: escaneando colección			8	9

- inicializando: createIndex está preparando el generador de índices. Esta fase debe ser muy breve.
- creando índice: inicializando: el generador de índices se está preparando para crear el índice. Esta fase debe ser muy breve.
- creando índice: escaneando colección: el generador de índices está escaneando la colección para recopilar las claves del índice. La unidad de medida son los “bloques”.

> **Note**
>
> Si hay más de un elemento de trabajo configurado para la compilación del índice, se mostrará en esta etapa. La etapa “escaneando colección” es la única etapa en la que se utilizan varios procesos de trabajo durante el proceso de compilación del índice. En todas las demás etapas se mostrará un único proceso de trabajo.

- creando índice: clasificando claves 1: el generador de índices está clasificando las claves de índice recopiladas. La unidad de medida son las “claves”.
- creando índice: clasificando claves 2: el generador de índices está clasificando las claves de índice recopiladas que corresponden a tuplas muertas. Esta fase solo existe para la compilación de índices únicos. La unidad de medida son las “claves”.

- creando índice: insertando claves: el generador de índices está insertando claves de índice en el nuevo índice. La unidad de medida son las “claves”.
- validando: escaneando índice: createIndex está escaneando el índice para buscar las claves que deben validarse. La unidad de medida son los “bloques”.
- validando: clasificando tuplas: createIndex está ordenando el resultado de la fase de escaneo del índice.
- validando: escaneando colección: createIndex está escaneando la colección para validar las claves de índice encontradas en las dos fases anteriores. La unidad de medida son los “bloques”.

Ejemplo de salida de compilación del índice

En el siguiente ejemplo de salida (creación de índices en primer plano), se muestra el estado de la creación del índice. El campo “msg” resume el progreso de la compilación indicando la etapa y el porcentaje de finalización de la compilación. El campo “procesos de trabajo” indica el número de procesos de trabajo utilizados durante esa etapa de la compilación del índice. El campo “progreso” muestra los números reales utilizados para calcular el porcentaje de finalización.

> **Note**
>
> Los campos «currentIndexBuildName», «msg» y «progress» no son compatibles con la versión 4.0 de Amazon DocumentDB.

```
{
    "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
}
```

Mantenimiento de índices de Amazon DocumentDB

Temas

- Sobrecarga del índice
- Mantenimiento de índices mediante ReIndex

Sobrecarga del índice

Amazon DocumentDB usa el control de Multi-Version concurrencia (MVCC) para administrar las transacciones simultáneas. Cuando los documentos se eliminan o actualizan, sus versiones anteriores permanecen en las colecciones e índices como versiones «inactivas». El proceso de recopilación de elementos no utilizados recupera automáticamente el espacio de estas versiones inactivas para futuras operaciones.

La sobrecarga de índices se produce cuando el tamaño de los índices de una colección aumenta debido a la acumulación de índices inactivos u obsoletos o a la fragmentación de las páginas. El porcentaje informado representa la cantidad de espacio de índice que pueden utilizar las entradas de índice futuras. Esta sobrecarga consume espacio tanto en la memoria caché como en el almacenamiento del búfer. Si desea eliminar la sobrecarga, tendrá que recopilar los índices.

Example Ejemplo

Ejecute el siguiente comando para determinar el almacenamiento no utilizado del índice:

```
db.coll.aggregate({$indexStats:{}});
```

El resultado tendrá un aspecto parecido al siguiente:

```
{
    "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
    }
}
```

Puede reconstruir los índices sin tiempo de inactividad mediante el comando `reIndex`, que requiere escanear toda la colección. Consulte Mantenimiento de índices mediante ReIndex .

Mantenimiento de índices mediante ReIndex

`reIndex` es un comando que se utiliza para reconstruir un índice. Se utiliza normalmente cuando un índice está dañado o es ineficiente. Con el tiempo, los índices pueden acumular espacio no utilizado debido a muchas actualizaciones, inserciones o eliminaciones, lo que reduce el rendimiento. La reindexación ayuda a eliminar ese espacio no utilizado y a restablecer la eficiencia del índice.

Pautas de ReIndex

- `reIndex` solo es compatible con Amazon DocumentDB 5.0.
- Amazon DocumentDB admite `reindex` un único índice en segundo plano, lo que permite varios procesos de trabajo. Las consultas pueden utilizar el índice anterior cuando el proceso de `reIndex` está en ejecución.

- Amazon DocumentDB admite informes de progreso de indexación a través de `currentOp`. Puede ver las etapas de creación del índice de forma similar a las Etapas de compilación del índice vistas durante la creación del índice. La única diferencia es que `reIndex` siempre tiene ocho etapas, independientemente de si son únicas o no. No existe una etapa de “creación de índices: claves de clasificación 2”.
- `reIndex` se puede ejecutar simultáneamente con cualquier comando, excepto los comandos relacionados con índices de la misma colección: `createIndexes`, `dropIndexes`, `collMod` y `renameCollection`.
- `reIndex` actualmente no es compatible con índices de texto, geoespaciales, vectoriales ni parciales.

Creación de ReIndex

Use el siguiente comando para reconstruir el índice:

```
db.runCommand({ reIndex: "collection-name", index: "index-name"})
```

Si lo desea, también puede controlar el número de procesos de trabajo asignados al proceso de reconstrucción:

```
db.runCommand({ reIndex: "collection-name", index: "index-name", workers: number })
```

Gestión de la compresión de documentos a nivel de colección

La compresión de documentos a nivel de colección de Amazon DocumentDB 5.0 le permite reducir los costos de almacenamiento y E/S al comprimir los documentos de sus colecciones. Puede habilitar la compresión de documentos a nivel de colección y ver las métricas de compresión según sea necesario, midiendo las ganancias de almacenamiento mediante métricas de compresión como es el tamaño de almacenamiento de los documentos comprimidos y el estado de la compresión. Amazon DocumentDB 5.0 utiliza el algoritmo de compresión LZ4 para comprimir documentos.

Amazon DocumentDB admite la compresión de documentos a partir de la versión 5.0. Las siguientes son funciones de compresión de documentos a nivel de colección:

- Comportamiento predeterminado: la configuración de compresión predeterminada para las nuevas colecciones de un clúster 5.0 viene determinada por el parámetro del clúster.

`default_collection_compression` Este parámetro está establecido en “deshabilitado” de forma predeterminada.

- Compresión de las colecciones existentes: la configuración de compresión de las colecciones existentes se puede cambiar mediante el comando `collMod`.
- Cambio del umbral de compresión: el umbral de compresión predeterminado es de 2 KB. Este valor se puede especificar para las colecciones nuevas mediante el comando `createCollection` y se puede cambiar para las colecciones existentes mediante el comando `collMod`.

> **Note**
>
> La compresión de documentos de Amazon DocumentDB no es compatible en las versiones 3.6 y 4.0 de Amazon DocumentDB.

Temas

- Administración de la compresión de documentos
- Supervisión de la compresión de documentos

Administración de la compresión de documentos

Habilitación de la compresión de documentos en una colección

Active la compresión de documentos al crear una colección en Amazon DocumentDB 5.0 mediante `db.createCollection()` el método:

```
db.createCollection( sample_collection,{
    storageEngine : {
        documentDB: {
            compression:{enable: <true | false>}
        }
    }
})
```

Habilitación de la compresión de documentos en un clúster

La compresión de documentos se puede habilitar de forma predeterminada para todas las colecciones nuevas a nivel de clúster al establecer el parámetro `default_collection_compression` del clúster en “habilitado”. Si este parámetro se establece en “habilitado”, las colecciones recién creadas en el clúster tendrán la compresión habilitada de forma predeterminada con un umbral de compresión de 2 KB.

Compresión de colecciones existentes

También puede modificar la configuración de compresión de una colección existente mediante la operación `collMod` y especificando la siguiente configuración `storageEngine`. Tenga en cuenta que el cambio realizado con este comando solo se aplicará a inserted/updated los documentos nuevos y la compresión de los documentos insertados anteriormente no cambiará.

```
db.runCommand({
    collMod: "orders",
    storageEngine: {
        documentDB: {compression: {enable: <true | false>} }
    }
})
```

Establecimiento de los umbrales de compresión

De forma predeterminada, el umbral de compresión de las colecciones comprimidas es de 2032 bytes. Este valor de umbral se puede establecer en el comando `createCollection` al crear una nueva colección con la compresión habilitada:

```
db.createCollection( sample_collection, {
    storageEngine : {
        documentDB: {
            compression: {
                enable: true,
                threshold: <128 - 8000>
            }
        }
    }
})
```

También puede modificar el umbral de compresión de una colección comprimida existente mediante la operación `collMod` y al especificar la siguiente configuración `storageEngine`:

```
db.runCommand({
    collMod: "orders",
    storageEngine: {
        documentDB: {
            compression: {
                enable: true,
                threshold: <128 - 8000>
            }
        }
    }
})
```

Tenga en cuenta que el umbral de compresión solo se puede establecer en un valor entre 128 y 8000 bytes. Además, la opción `enable` debe estar configurada en “true” al especificar el umbral de compresión.

Supervisión de la compresión de documentos

Puede comprobar si una colección está comprimida y calcular la relación de compresión de la siguiente manera:

Para ver las estadísticas de compresión, ejecute el comando `db.printCollectionStats()` o `db.collection.stats()` desde el intérprete de comandos de mongo. El resultado muestra el tamaño original y el tamaño comprimido, que puede comparar para analizar las ganancias de almacenamiento derivadas de la compresión de documentos. En este ejemplo, se muestran las estadísticas de una colección denominada “sample_collection” a continuación. A continuación, se utiliza un factor de escala de 1024*1024 para generar los valores `size` y `storageSize` en MB.

```
db.sample_collection.stats(1024*1024)
```

El siguiente es un ejemplo del resultado para el comando anterior:

```
{
    "ns" : "test.sample_collection",
    "count" : 1000000,
    "size" : 3906.3,
    "avgObjSize" : 4096,
    "storageSize" : 1953.1,
    compression:{"enabled" : true,"threshold" : 2032},
    ...
```

```
}
```

- tamaño: el tamaño original de la colección de documentos.
- promedioObjSize: el tamaño medio del documento antes de la compresión se redondea al primer decimal. La unidad de medida son bytes.
- StorageSize: el tamaño de almacenamiento de la colección después de la compresión. La unidad de medida son bytes.
- habilitada: indica si la compresión está habilitada o deshabilitada.

Para calcular la relación de compresión real, divida el tamaño de la colección por el tamaño de almacenamiento (size/storageSize). Para el ejemplo anterior, el cálculo es 3906. 3/1953.1, lo que se traduce en una relación de compresión de 2:1.

Administración de la compresión basada en diccionarios en Amazon DocumentDB 8.0

Amazon DocumentDB 8.0 presenta un nuevo algoritmo de compresión de documentos, zstd, como una alternativa mejorada a LZ4. Al habilitar la compresión de diccionarios en una colección de Amazon DocumentDB 8.0 seleccionando Zstd como algoritmo de compresión, los documentos de la colección se muestrean para crear un diccionario de compresión personalizado. Los documentos nuevos y actualizados se comprimen con este diccionario y el algoritmo zstd. Este enfoque suele lograr relaciones de compresión más altas que los métodos de compresión estándar, especialmente para colecciones con esquemas de documentos consistentes o nombres de campo repetidos.

Lz4 es un algoritmo diseñado para una compresión y descompresión rápidas. Se sabe que es liviano para la CPU y, al mismo tiempo, logra una compresión notable. Zstd es un algoritmo de uso general que, con la configuración predeterminada, usa más CPU pero logra mejores relaciones de compresión que lz4. El uso de diccionarios mejora aún más la compresión en la mayoría de los documentos JSON. Algunas de las ventajas del algoritmo Zstd son:

- Reducción de los costos de almacenamiento: las relaciones de compresión más altas significan menos uso de almacenamiento y menores costos.
- Menor I/O: los documentos comprimidos requieren menos I/O, lo que podría mejorar el rendimiento.
- Optimizado para su colección: el diccionario está diseñado específicamente para los patrones de datos de su colección.

> **Note**
>
> Dictionary-based la compresión no se admite en las versiones 3.6, 4.0 y 5.0 de Amazon DocumentDB.

Consideraciones sobre el rendimiento

La compresión Zstd implica las siguientes ventajas y desventajas:

- Almacenamiento frente a CPU: la compresión Zstd logra mejores relaciones de compresión, pero puede consumir un poco más de recursos de CPU en comparación con la compresión LZ4.
- Compresión inicial: es posible que las nuevas colecciones no logren una compresión óptima hasta que no se hayan insertado suficientes documentos para formar un diccionario eficaz. En la actualidad, un diccionario se entrena si reúne al menos 100 documentos.
- Tipo de carga de Read-intensive trabajo: las cargas de trabajo en las que todos los datos caben en la caché del búfer pueden experimentar un aumento de la latencia y del uso de la CPU debido a la sobrecarga de descompresión.

La compresión Zstd es especialmente eficaz para colecciones con documentos pequeños, matrices de documentos y nombres de campos repetidos.

Habilitar la compresión basada en diccionarios

En el caso de colecciones nuevas, puede utilizar el siguiente comando para activar la compresión Zstd:

```
db.createCollection("myCollection",
    {
        storageEngine: {
            documentDB: {
                compression: {
                    enable: true,
                    algorithm: "zstd"
                }
            }
        }
    }
```

```
)
```

También puedes activar o modificar la compresión en una colección existente:

```
db.runCommand({
    collMod: "myCollection",
    storageEngine: {
        documentDB: {
            compression: {
                enable: true,
                algorithm: "zstd"
            }
        }
    }
})
```

Para habilitar el algoritmo Zstd en todas las colecciones del clúster, puede modificar el grupo de parámetros del clúster para seleccionar «zstd» como valor para el parámetro «default_collection_compression».

Introducción

Amazon DocumentDB 8.0 incluye la compresión Zstd activada de forma predeterminada. Siempre puede desactivarla configurando el valor de 'default_compression' como deshabilitado en el grupo de parámetros del clúster. Debe tenerse en cuenta que, a partir de Amazon DocumentDB 8.0, «activado» ya no es una opción válida para default_compression y debe seleccionar entre Zstd y LZ4.

Supervisión

Puede ver la información de compresión de una colección mediante uno de los siguientes comandos:

- db.runCommand ({CollStats: «myCollection»}) O
- db.collection.stats ()

Estos comandos devuelven estadísticas clave que puede utilizar para calcular la relación de compresión:

- algoritmo de compresión: el algoritmo utilizado («lz4" o «zstd»)
- StorageSize: el almacenamiento real que utiliza la colección después de la compresión. Tenga en cuenta que este número incluye la fragmentación (es decir, el espacio no utilizado en las páginas de la base de datos)
- promedioObjSize: el tamaño lógico promedio de los documentos de la colección, descomprimido. Tenga en cuenta que si su colección tiene más de 20 000 documentos, este número será una aproximación basada en una muestra de 20 000 documentos.
- tamaño: el tamaño lógico de la colección sin compresión. Este número se obtiene multiplicando el promedio ObjSize por el número total de documentos de la colección, por lo que si el promedio ObjSize es una aproximación, este número también será una aproximación.
- recuento: número de documentos de la colección

Las siguientes CloudWatch métricas pueden resultar útiles a la hora de evaluar la compresión basada en diccionarios:

- CPUUtilization
- FreeableMemory
- VolumeBytesUsed
- VolumeReadIOPs
- VolumeWriteIOPs

Métricas de CollStats:

- storageSize
- tamaño

Además, puede resultar útil realizar un seguimiento de las métricas específicas de la aplicación, como la latencia y el rendimiento por tipo de consulta o API.

Seguridad en Amazon DocumentDB

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, se beneficia de una arquitectura de red y centro de datos diseñada para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre usted AWS y usted. Esta documentación le permite comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza Amazon DocumentDB. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta AWS los servicios en la AWS nube. AWS también le proporciona servicios que puede usar de forma segura. Third-party los auditores prueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los programas de AWS cumplimiento. Para obtener más información acerca de los programas de conformidad que se aplican a Amazon DocumentDB (compatible con MongoDB), consulte Servicios de AWS en el ámbito del programa de conformidad.
- Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice. Usted también es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos aplicables.

Amazon DocumentDB está autorizada por el Programa federal de administración de riesgos y autorizaciones (FedRAMP). Cuenta con una autorización alta de FedRAMP para AWS GovCloud las regiones de EE. UU. y una autorización moderada de FedRAMP para las regiones de EE. UU. AWS East/West Para obtener más información sobre los esfuerzos de cumplimiento AWS y las iniciativas de cumplimiento, consulte los AWS servicios según el programa de cumplimiento.

> **Note**
>
> Este capítulo se aplica a clústeres basados en instancias y clústeres elásticos. Para obtener más información, consulte los siguientes temas.

También aprenderá a utilizar otros AWS servicios que le ayudan a supervisar y proteger sus recursos de Amazon DocumentDB. En los siguientes temas, se mostrará cómo configurar Amazon DocumentDB para satisfacer sus objetivos de seguridad y conformidad.

Temas

- Administración de contraseñas con Amazon DocumentDB y AWS Secrets Manager
- Protección de datos en Amazon DocumentDB
- Identity and Access Management para Amazon DocumentDB
- Cross-service confusa prevención adjunta
- Autenticación mediante la identidad de IAM
- Autenticación mediante SCRAM-SHA-256
- Administración de usuarios de Amazon DocumentDB
- Acceso a bases de datos mediante Role-Based Access Control
- Actualización de los certificados TLS de Amazon DocumentDB
- Validación de la conformidad en Amazon DocumentDB
- Resiliencia de Amazon DocumentDB
- Seguridad de la infraestructura en Amazon DocumentDB
- Puntos finales de API e interfaz de VPC de Amazon DocumentDB (AWS PrivateLink)
- Prácticas recomendadas de seguridad para Amazon DocumentDB
- Auditoría de eventos de Amazon DocumentDB
- Amazon VPC y Amazon DocumentDB

Administración de contraseñas con Amazon DocumentDB y AWS Secrets Manager

Amazon DocumentDB se integra con Secrets Manager para administrar las contraseñas de los usuarios principales de los clústeres.

Temas

- Limitaciones de la integración de Secrets Manager con Amazon DocumentDB
- Descripción general de la administración de las contraseñas de los usuarios principales con AWS Secrets Manager
- Imponer a Amazon DocumentDB la administración de la contraseña del usuario principal en AWS Secrets Manager
- Administración de la contraseña del usuario principal para un clúster con Secrets Manager

Limitaciones de la integración de Secrets Manager con Amazon DocumentDB

Las siguientes características no admiten la administración de contraseñas del usuario principal con Secrets Manager:

- Clústeres que forman parte de una base de datos global de Amazon DocumentDB.
- Réplicas de lectura de Amazon DocumentDB para diferentes regiones.

Descripción general de la administración de las contraseñas de los usuarios principales con AWS Secrets Manager

Con AWS Secrets Manager, puedes reemplazar las credenciales codificadas de forma rígida en tu código, incluidas las contraseñas de las bases de datos, por una llamada a la API a Secrets Manager para recuperar el secreto mediante programación. Para obtener más información acerca de Secrets Manager, consulte la Guía del usuario de AWS Secrets Manager.

Cuando guardas secretos de bases de datos en Secrets Manager, tu AWS cuenta incurre en cargos. Para obtener información sobre precios, consulte Precios de AWS Secrets Manager.

Puede especificar que Amazon DocumentDB administre la contraseña de usuario principal en Secrets Manager para un clúster de Amazon DocumentDB cuando realice una de las siguientes operaciones:

- Cree el clúster
- Modificación del clúster

Cuando especifica que Amazon DocumentDB administre la contraseña del usuario principal en Secrets Manager, Amazon DocumentDB genera la contraseña y la almacena en Secrets Manager. Puede interactuar directamente con el secreto para recuperar las credenciales del usuario principal. También puedes especificar una clave gestionada por el cliente para cifrar el secreto o usar la clave de KMS que proporciona Secrets Manager.

Amazon DocumentDB administra la configuración del secreto y lo rota cada siete días de forma predeterminada. Puede modificar algunos de los ajustes, como el programa de rotación. Si elimina un clúster que administra un secreto en Secrets Manager, también se eliminarán el secreto y sus metadatos asociados.

Para conectarse a un clúster con las credenciales en un secreto, puede recuperar el secreto en Secrets Manager. Para obtener más información, consulte Obtener datos secretos de una base de datos SQL AWS Secrets Manager y conectarse a ella mediante JDBC con credenciales en AWS Secrets Manager secreto en la Guía del AWS Secrets Manager usuario.

Imponer a Amazon DocumentDB la administración de la contraseña del usuario principal en AWS Secrets Manager

Puede utilizar las claves de condición de IAM para hacer que Amazon DocumentDB administre la contraseña del usuario principal en AWS Secrets Manager. La siguiente política no permite que los usuarios creen ni restauren instancias ni clústeres a menos que Amazon DocumentDB administre la contraseña del usuario principal en Secrets Manager.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "rds:CreateDBCluster"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "rds:ManageMasterUserPassword": false
                }
            }
        }
    ]
}
```

Administración de la contraseña del usuario principal para un clúster con Secrets Manager

Para configurar la administración de Amazon DocumentDB de la contraseña del usuario principal en Secrets Manager, debe realizar las siguientes acciones:

- Creación de un clúster de Amazon DocumentDB
- Modificación de un clúster de Amazon DocumentDB

Puede utilizar la consola de Amazon DocumentDB o la AWS CLI para realizar estas acciones.

Protección de datos en Amazon DocumentDB

El modelo de responsabilidad AWS https://aws.amazon.com/compliance/shared-responsibility-model/ se aplica a la protección de datos en Amazon DocumentDB. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta todo el Nube de AWS. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. También eres responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulte las Preguntas frecuentes sobre la privacidad de los datos y las condiciones legales de la región de . Para obtener más información sobre la protección de datos en Europa, consulte el Centro del Reglamento General de Protección de Datos (RGPD).

Con fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure los usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Úselo SSL/TLS para comunicarse con AWS los recursos. Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Configure el registro de actividad de los usuarios y de la API con AWS CloudTrail. Para obtener información sobre el uso de CloudTrail rutas para capturar AWS actividades, consulte Cómo trabajar con CloudTrail rutas en la Guía del AWS CloudTrail usuario.
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados que contiene Servicios de AWS.
- Utiliza servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger la información confidencial almacenada en Amazon S3.
- Si necesita módulos criptográficos validados por la FIPS 140-3 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información

sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-3.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabaja con Amazon DocumentDB u otro tipo de software Servicios de AWS mediante la consola, la API o los SDK. AWS CLI AWS Cualquier dato que introduzca en etiquetas o campos de formato libre utilizados para los nombres se pueden emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Temas

- Client-side cifrado a nivel de campo
- Cifrado de datos en reposo de Amazon DocumentDB
- Cifrado de datos en tránsito
- Administración de claves

Client-side cifrado a nivel de campo

El cifrado a nivel de campo (FLE) del lado del cliente de Amazon DocumentDB le permite cifrar la información confidencial de las aplicaciones cliente antes de transferirlos a un clúster de Amazon DocumentDB. La información confidencial permanece cifrada cuando se almacenan y procesan en un clúster y se descifran en la aplicación cliente cuando se recuperan.

Temas

- Introducción
- Consultas en el FLE del lado del cliente

Introducción

La configuración inicial del FLE del lado del cliente en Amazon DocumentDB es un proceso de cuatro pasos que incluye la creación de una clave de cifrado, la asociación de un rol a la aplicación, la configuración de la aplicación y la definición de la operación CRUD con las opciones de cifrado.

Temas

- Paso 1: crear las claves de cifrado
- Paso 2: asociar un rol a la aplicación
- Paso 3: configurar la aplicación
- Paso 4: definir una operación CRUD
- Ejemplo: archivo de configuración de cifrado a nivel de campo del lado del cliente

Paso 1: crear las claves de cifrado

Con él AWS Key Management Service, cree una clave simétrica gestionada por el cliente que cifre y descifre el campo de datos confidenciales y concédale los permisos de uso de IAM necesarios. AWS KMS almacena la clave gestionada por el cliente, que se utiliza para cifrar las claves de datos. Almacenar la clave gestionada por el cliente AWS KMS refuerza su postura de seguridad. La clave de datos es la clave secundaria, que se almacena en una colección de Amazon DocumentDB y es necesaria para cifrar los campos confidenciales antes de almacenar el documento en Amazon DocumentDB. La clave gestionada por el cliente cifra la clave de datos, que a su vez cifra y descifra sus datos. Si usas un clúster global, puedes crear una clave multirregional que los distintos roles de servicio puedan usar en diferentes regiones.

Para obtener más información sobre la clave AWS Key Management Service, incluida la forma de crear una clave, consulta AWS Key Management Service los conceptos en la Guía para AWS Key Management Service desarrolladores.

Paso 2: asociar un rol a la aplicación

Cree una política de IAM con AWS KMS los permisos adecuados. Esta política permite a las identidades de IAM a las que está asociada cifrar y descifrar la clave de KMS especificada en el campo de recursos. La aplicación asume esta función de IAM con la que se autentica. AWS KMS

El aspecto de la respuesta debe ser parecido al siguiente:

```
{ "Effect": "Allow",
"Action": ["kms:Decrypt", "kms:Encrypt"],
"Resource": "Customer Key ARN"
}
```

Paso 3: configurar la aplicación

Ha definido una clave gestionada por el cliente AWS KMS y ha creado una función de IAM con los permisos necesarios para acceder a ella. Importe estos paquetes necesarios.

```
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. Especifique `aws` el tipo AWS KMS de proveedor e introduzca las credenciales de la cuenta que recuperó en el paso anterior.

   ```
   provider = "aws"
   kms_providers = {
       provider: {
           "accessKeyId": current_credentials.access_key,
           "secretAccessKey": current_credentials.secret_key
       }
   }
   ```

2. Especifique la clave gestionada por el cliente para cifrar la clave de datos:

   ```
   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. Configure el MongoClient objeto:

   ```
   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. Genere su clave de datos:

```
data_key_id = client_encryption.create_data_key(provider,
    customer_key,
    key_alt_name = [key_alt_name])
```

5. Recupera tu clave de datos existente:

```
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})
```

Paso 4: definir una operación CRUD

Defina la operación CRUD con opciones de cifrado.

1. Defina la colección para write/read /eliminar un solo documento:

```
coll = client.gameinfo.users
```

2. Cifrado explícito: cifra los campos e inserta:

> **Note**
>
> Debe proporcionarse exactamente uno de los siguientes valores: “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"]
})
```

Ejemplo: archivo de configuración de cifrado a nivel de campo del lado del cliente

Para utilizar el ejemplo siguiente, sustituya *placeholder* con su propia información.

```
# 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()
```

Consultas en el FLE del lado del cliente

Amazon DocumentDB solo admite consultas de igualdad con valores cifrados con FLE del lado del cliente.

Por ejemplo, para consultar documentos en los que la puntuación de jugador cifrada sea igual a 500, el cliente utiliza un método de cifrado explícito para cifrar el valor de la consulta:

```
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 }
} )
```

Cifrado de datos en reposo de Amazon DocumentDB

> **Note**
>
> AWS KMS está sustituyendo el término clave maestra del cliente (CMK) por clave KMS. AWS KMS key El concepto no ha cambiado. Para evitar cambios importantes, AWS KMS mantiene algunas variaciones de este término.

Puede cifrar los datos en reposo del clúster de Amazon DocumentDB especificando la opción de cifrado de almacenamiento al crear el clúster. El cifrado de almacenamiento está habilitado para todo el clúster y se aplica a todas las instancias, incluida la instancia principal y las réplicas. También se aplica al volumen de almacenamiento, los datos, los índices, los registros, las copias de seguridad automatizadas y las instantáneas del clúster.

Amazon DocumentDB utiliza el estándar de cifrado avanzado (AES-256) de 256 bits para cifrar los datos mediante las claves de cifrado almacenadas en (). AWS Key Management Service AWS KMS Si utiliza un clúster de Amazon DocumentDB con el cifrado en reposo activado, no necesita modificar la lógica de la aplicación ni la conexión del cliente. Amazon DocumentDB se encarga del cifrado y descifrado de sus datos de forma transparente con un impacto mínimo en el desempeño.

Amazon DocumentDB se integra AWS KMS y utiliza un método conocido como cifrado de sobres para proteger sus datos. Cuando un clúster de Amazon DocumentDB se cifra con un AWS KMS, Amazon DocumentDB solicita el uso de su clave de KMS AWS KMS para generar una clave de datos de texto cifrado a fin de cifrar el volumen de almacenamiento. La clave de datos de texto cifrado se cifra mediante la clave KMS que ha definido y se almacena junto con los datos cifrados y los metadatos de almacenamiento. Cuando Amazon DocumentDB necesita acceder a sus datos cifrados, solicita descifrar la clave de datos de texto cifrado AWS KMS con su clave de KMS y almacena en caché la clave de datos de texto sin formato en la memoria para cifrar y descifrar de manera eficiente los datos del volumen de almacenamiento.

La función de cifrado de almacenamiento de Amazon DocumentDB está disponible para todos los tamaños de instancia admitidos y en todos los lugares en los que Amazon DocumentDB esté disponible. Regiones de AWS

Habilitación del cifrado en reposo de un clúster de Amazon DocumentDB

Puede habilitar o deshabilitar el cifrado en reposo en un clúster de Amazon DocumentDB cuando el clúster se aprovisione mediante el comando Consola de administración de AWS o el (). AWS Command Line Interface AWS CLI Los clústeres creados con la consola tienen el cifrado en reposo habilitado de forma predeterminada. Los clústeres que cree con el AWS CLI tienen el cifrado en reposo deshabilitado de forma predeterminada. Por lo tanto, debe habilitar explícitamente el cifrado en reposo mediante el `--storage-encrypted` parámetro. En cualquier caso, una vez creado el clúster, no puede cambiar la opción de cifrado en reposo.

Amazon DocumentDB se utiliza AWS KMS para recuperar y administrar las claves de cifrado y para definir las políticas que controlan la forma en que se pueden usar estas claves. Si no especifica un identificador de AWS KMS clave, Amazon DocumentDB utiliza la clave de KMS del servicio AWS gestionado predeterminada. Amazon DocumentDB crea una clave de KMS independiente para cada Región de AWS una de sus. Cuenta de AWS Para obtener más información, consulte Conceptos de AWS Key Management Service.

Para comenzar a crear su propia KMS, consulte Introducción en la AWS Key Management Service Guía del desarrollador.

> **Important**
>
> Debe utilizar una clave de cifrado de KMS simétrica para cifrar el clúster, ya que Amazon DocumentDB solo admite claves de cifrado de KMS de cifrado simétricas. No utilice una KMS asimétrica para intentar cifrar los datos de los clústeres de Amazon DocumentDB. Para obtener más información, consulte claves asimétricas de AWS KMS en la Guía para desarrolladores de AWS Key Management Service .

Si Amazon DocumentDB ya no puede acceder a la clave de KMS de un clúster (por ejemplo, cuando el Cuenta de AWS propietario de la clave está suspendido, la clave está deshabilitada, la clave está programada para su eliminación o se elimina la política o concesión de claves en la que se basa Amazon DocumentDB), el clúster pasa primero al estado. `inaccessible-encryption-credentials-recoverable` Mientras el clúster se encuentra en este estado, Amazon DocumentDB detiene las instancias del clúster y no puede leer ni escribir en el clúster,

pero el clúster aún se puede recuperar si se restaura el acceso a la clave de KMS en un plazo de 7 días. Si el acceso no se restablece en un plazo de 7 días, el clúster pasa al `inaccessible-encryption-credentials` estado de terminal. Desde el estado terminal, el clúster ya no está disponible y el estado actual de la base de datos no se puede recuperar; solo se puede restaurar a partir de una copia de seguridad o realizar una restauración puntual con la clave de KMS original. En Amazon DocumentDB, las copias de seguridad siempre están habilitadas durante al menos 1 día.

> **Note**
>
> Los clústeres que forman parte de un clúster global se comportan de forma diferente. Cuando Amazon DocumentDB detecta que ya no puede acceder a la clave de KMS, todos los clústeres del clúster global pasan directamente al estado de terminal y omiten el `inaccessible-encryption-credentials` estado recuperable. Esto se debe a que un clúster que forma parte de un clúster global solo se puede detener e iniciar cuando es el único clúster del clúster global. Para realizar la recuperación, debe realizar una restauración a partir de una instantánea o realizar una restauración puntual. Para eliminar los clústeres originales, primero debe eliminar cada clúster del clúster global y, a continuación, eliminarlo.

> **Important**
>
> No puede cambiar la clave KMS de un clúster cifrado después de haberlo creado. Asegúrese de determinar los requisitos de clave de cifrado antes de crear el clúster cifrado.

Using the Consola de administración de AWS

Debe especificar la opción de cifrado en reposo al crear un clúster. El cifrado en reposo se habilita de forma predeterminada cuando se crea un clúster mediante la Consola de administración de AWS. No se puede cambiar una vez creado el clúster.

Para especificar la opción de cifrado en reposo, al crear el clúster

1. Cree un clúster de Amazon DocumentDB como se describe en la sección Introducción. Sin embargo, en el paso 6, no elija Create cluster (Crear clúster).
2. Debajo de la sección Authentication (Autenticación), elija Show advanced settings (Mostrar configuración avanzada).
3. Desplácese hacia abajo hasta la sección de la Encryption-at-rest.

4. Elija la opción que desee para el cifrado en reposo. Cualquiera que sea la opción que elija, no podrá cambiarla después de crear el clúster.
   - Para cifrar datos en reposo en este clúster, elija Enable encryption (Habilitar cifrado).
   - Si no desea cifrar datos en reposo en este clúster, elija Disable encryption (Deshabilitar cifrado).
5. Elija la clave principal que desee. Amazon DocumentDB usa el AWS Key Management Service (AWS KMS) para recuperar y administrar las claves de cifrado y para definir las políticas que controlan la forma en que se pueden usar estas claves. Si no especifica un identificador de AWS KMS clave, Amazon DocumentDB utiliza la clave KMS del servicio AWS gestionado predeterminada. Para obtener más información, consulte Conceptos de AWS Key Management Service.

   > **Note**
   >
   > Después de crear un clúster cifrado, no puede cambiar la clave KMS de dicho clúster. Asegúrese de determinar los requisitos de clave de cifrado antes de crear el clúster cifrado.

6. Rellene las demás secciones según sea necesario y cree el clúster.

Using the AWS CLI

Para cifrar un clúster de Amazon DocumentDB mediante AWS CLI, ejecute el `create-db-cluster` comando y especifique la opción. `--storage-encrypted` Los clústeres de Amazon DocumentDB creados con la opción AWS CLI No habilita el cifrado de almacenamiento de forma predeterminada.

En el siguiente ejemplo se crea un clúster de Amazon DocumentDB con el cifrado de almacenamiento habilitado.

En los siguientes ejemplos, sustituya cada uno por *`user input placeholder`* la información de su clúster.

Example

Para 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
```

Para Windows:

```
aws docdb create-db-cluster ^
  --db-cluster-identifier SampleUser1 ^
  --port 27017 ^
  --engine docdb ^
  --master-username SampleUser1 ^
  --master-user-password primaryPassword ^
  --storage-encrypted
```

Al crear un clúster de Amazon DocumentDB cifrado, puede especificar un identificador de AWS KMS clave, como en el ejemplo siguiente.

Example

Para 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
```

Para 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**
>
> Después de crear un clúster cifrado, no puede cambiar la clave KMS de dicho clúster. Asegúrese de determinar los requisitos de clave de cifrado antes de crear el clúster cifrado.

Resolver un clúster de Amazon DocumentDB en un estado de cifrado inaccesible

Un clúster de Amazon DocumentDB pasa a un estado de cifrado inaccesible cuando Amazon DocumentDB no puede acceder a la clave de KMS con la que se cifró el clúster. Existen dos estados de este tipo y la ruta de recuperación depende de en qué estado se encuentre el clúster.

`estado inaccesible: cifrado, credenciales recuperables`

Mientras el clúster esté en ese `inaccessible-encryption-credentials-recoverable` estado, puede devolverlo a: para ello, restaure el acceso de Amazon DocumentDB a la clave de `available` KMS y, a continuación, inicie el clúster. Para resolver este estado, haga lo siguiente:

1. Confirme Cuenta de AWS que el propietario de la clave de KMS esté activo. Si la cuenta está suspendida, vuelva a activarla.
2. Confirme que la clave KMS esté habilitada. Para obtener más información, consulte Habilitación y deshabilitación de claves en la Guía para desarrolladores de AWS Key Management Service .
3. Compruebe si la clave de KMS está programada para su eliminación. Si es así, cancele la eliminación programada de la clave. Para obtener más información, consulte Programar y cancelar la eliminación de claves en la Guía para AWS Key Management Service desarrolladores.
4. Confirme que la política de claves de KMS y cualquier concesión de la que dependa Amazon DocumentDB sigan permitiendo a Amazon DocumentDB utilizar la clave.
5. Una vez restablecido el acceso a la clave de KMS, inicie el clúster mediante el comando Consola de administración de AWS o ejecutando el `start-db-cluster` AWS CLI comando.

   Example

   Para Linux, macOS o Unix:

   ```
   aws docdb start-db-cluster \
     --db-cluster-identifier example-cluster
   ```

Para Windows:

```
aws docdb start-db-cluster ^
  --db-cluster-identifier example-cluster
```

> **Important**
>
> Si el acceso a la clave de KMS no se restablece en un plazo de 7 días, el clúster pasa al `inaccessible-encryption-credentials` estado de terminal, desde el que no se puede iniciar.

`estado de credenciales de cifrado inaccesibles`

El estado es terminal. `inaccessible-encryption-credentials` No se puede iniciar el clúster y no se puede recuperar el estado de ejecución de la base de datos. Para recuperar los datos, restaure a partir de una instantánea o realice una restauración puntual en un nuevo clúster. Debe seguir teniendo acceso a la clave de KMS original para realizar la restauración. Si se eliminó la clave de KMS, los datos no se pueden recuperar.

Para obtener más información, consulte Restauración de una instantánea del clúster y Restauración a un momento dado.

> **Note**
>
> Los clústeres que forman parte de un clúster global pasan directamente al `inaccessible-encryption-credentials` estado en el que se pierde el acceso a la clave de KMS, ya que un clúster que forma parte de un clúster global solo se puede detener e iniciar cuando es el único clúster del clúster global. Para eliminar un clúster que se encuentre en este estado, primero elimine cada clúster del clúster global y, a continuación, elimine los clústeres de forma individual.

Si no puedes eliminar un clúster que se encuentra en ese `inaccessible-encryption-credentials` estado porque la protección contra la eliminación está habilitada, desactiva la protección contra la eliminación utilizando la opción AWS CLI antes de volver a intentar eliminarla.

Example

Para Linux, macOS o Unix:

```
aws docdb modify-db-cluster \
  --db-cluster-identifier example-cluster \
  --no-deletion-protection
```

Para Windows:

```
aws docdb modify-db-cluster ^
  --db-cluster-identifier example-cluster ^
  --no-deletion-protection
```

A continuación, puedes eliminar el clúster mediante el `delete-db-cluster` comando.

Example

Para Linux, macOS o Unix:

```
aws docdb delete-db-cluster \
  --db-cluster-identifier example-cluster \
  --skip-final-snapshot
```

Para Windows:

```
aws docdb delete-db-cluster ^
  --db-cluster-identifier example-cluster ^
  --skip-final-snapshot
```

Si el clúster no se elimina después de ejecutar los comandos anteriores, póngase en contacto con AWS Support.

Limitaciones para clústeres cifrados de Amazon DocumentDB

Los clústeres cifrados de Amazon DocumentDB tienen las siguientes limitaciones:

- Solo puede habilitar o deshabilitar el cifrado en reposo para un clúster de Amazon DocumentDB en el momento en que se crea, no después de que el clúster se haya creado. Sin embargo, puede crear una copia cifrada de un clúster sin cifrar creando una instantánea del clúster sin cifrar y,

a continuación, restaurando la instantánea sin cifrar como nuevo clúster mientras especifica la opción de cifrado en reposo.

Para obtener más información, consulte los temas siguientes:

- Creación de una instantánea manual del clúster
- Restauración de una instantánea del clúster
- Copia de instantáneas de clústeres de Amazon DocumentDB

- Los clústeres de Amazon DocumentDB con el cifrado de almacenamiento habilitado no se pueden modificar para deshabilitar el cifrado.
- Todas las instancias, copias de seguridad automatizadas, instantáneas e índices de un clúster de Amazon DocumentDB se cifran con la misma clave KMS.

Cifrado de datos en tránsito

Puede utilizar seguridad de la capa de transporte (TLS) para cifrar la conexión entre su aplicación y un clúster de Amazon DocumentDB. De forma predeterminada, el cifrado en tránsito está habilitado para los clústeres de Amazon DocumentDB recién creados. Opcionalmente se puede deshabilitar cuando se crea el clúster o en un momento posterior. Cuando se habilita el cifrado en tránsito, se requieren conexiones seguras con TLS para conectarse al clúster. Para obtener más información sobre cómo conectarse a Amazon DocumentDB mediante TLS, consulte Conexión mediante programación a Amazon DocumentDB.

Administración de la configuración de TLS del clúster de Amazon DocumentDB

El cifrado en tránsito para un clúster de Amazon DocumentDB se administra mediante el parámetro TLS en un grupo de parámetros de clúster. Puede administrar la configuración de TLS del clúster de Amazon DocumentDB mediante Consola de administración de AWS o AWS Command Line Interface ()AWS CLI. Consulte las secciones siguientes para obtener información sobre cómo verificar y modificar la configuración actual de TLS.

Using the Consola de administración de AWS

Siga estos pasos para realizar tareas de administración para el cifrado TLS mediante la consola, como identificar grupos de parámetros, verificar el valor de TLS y realizar las modificaciones necesarias.

> **Note**
>
> A menos que lo especifique de manera diferente al crear un clúster, el clúster se crea con el grupo de parámetros de clúster predeterminado. Los parámetros del grupo de parámetros del `default` clúster no se pueden modificar (por ejemplo, `tls` enabled/disabled). Por lo tanto, si su clúster está utilizando un `default` grupo de parámetros de clúster, debe modificar el clúster para utilizar un grupo de parámetros de clúster no predeterminado. En primer lugar, es posible que tenga que crear un grupo de parámetros de clúster personalizado. Para obtener más información, consulte Creación de grupos de parámetros de clúster de Amazon DocumentDB.

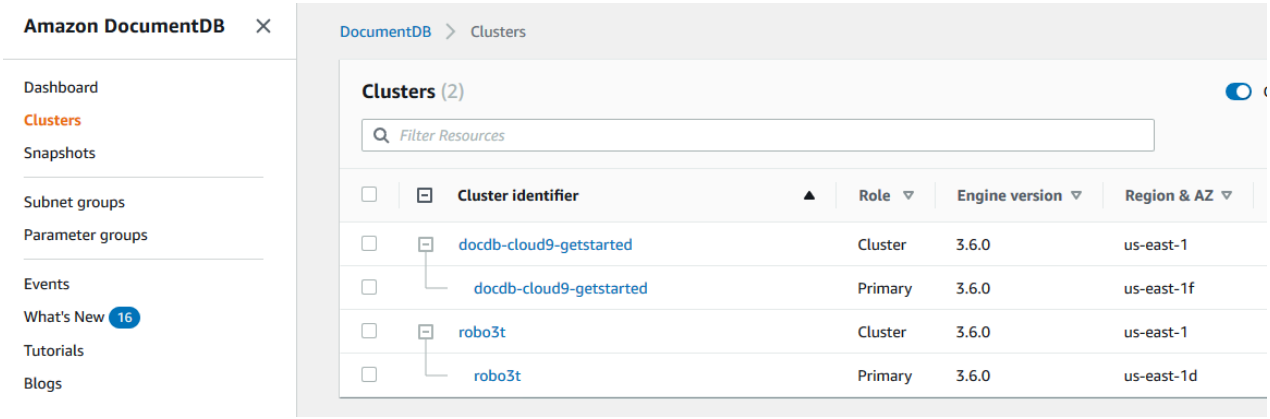
1. Determine el grupo de parámetros de clúster que utiliza el clúster.

   a. Abra la consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.

   b. En el panel de navegación, seleccione Clusters (Clústeres).

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

   c. Tenga en cuenta que en el cuadro de navegación Clústeres, la columna Identificador del clústeres muestra tanto los clústeres como las instancias. Las instancias se muestran debajo de los clústeres. Consulte la siguiente captura de pantalla como referencia.



   d. Elija el clúster que le interese.

e. Elija la pestaña de Configuración, desplácese hacia abajo hasta la parte inferior de Detalles del clúster) y localice el Grupo de parámetros de clúster. Anote el nombre del grupo de parámetros de clúster.

   Si el nombre del grupo de parámetros de clúster es `default` (por ejemplo, `default.docdb3.6`), debe crear un grupo de parámetros de clúster personalizado y convertirlo en el grupo de parámetros del clúster antes de continuar. Para obtener más información, consulte los siguientes temas:

   1. Creación de grupos de parámetros de clúster de Amazon DocumentDB: si no dispone de un grupo de parámetros de clúster personalizado que pueda utilizar, créelo.
   2. Modificación de un clúster de Amazon DocumentDB: modifique el clúster de forma que use el grupo de parámetros de clúster personalizado.

2. Determinar el valor actual del**`tls`** parámetro de clúster.

   a. Abra la consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.

   b. En el panel de navegación, seleccione Grupos de parámetros.

   c. En la lista de grupos de parámetros de clúster, elija el nombre del grupo de parámetros de clúster que desee.

   d. Localice la sección Cluster parameters (Parámetros de clúster). En la lista de parámetros de clúster, localice la `tls` fila del parámetro del clúster. En este punto, las cuatro columnas siguientes son importantes:

      - Nombre del parámetro de clúster: el nombre de los parámetros de clúster. Para la administración de TLS, le interesa el `tls` parámetro de clúster.
      - Valores: el valor actual de cada parámetro del clúster.
      - Valores permitidos: lista de valores que se pueden aplicar a un parámetro de clúster.
      - Tipo de aplicación: estática o dinámica. Los cambios en los parámetros de clúster estáticos solo se pueden aplicar al reiniciar las instancias. Los cambios en los parámetros de clúster dinámicos se pueden aplicar inmediatamente o al reiniciar las instancias.

3. Modificar el valor del **`tls`** parámetro de clúster.

   Si el valor de `tls` no es el que necesita, modifíquelo para este grupo de parámetros de clúster. Para cambiar el valor del `tls` parámetro de clúster, continúe desde la sección anterior siguiendo los pasos que se indican a continuación.

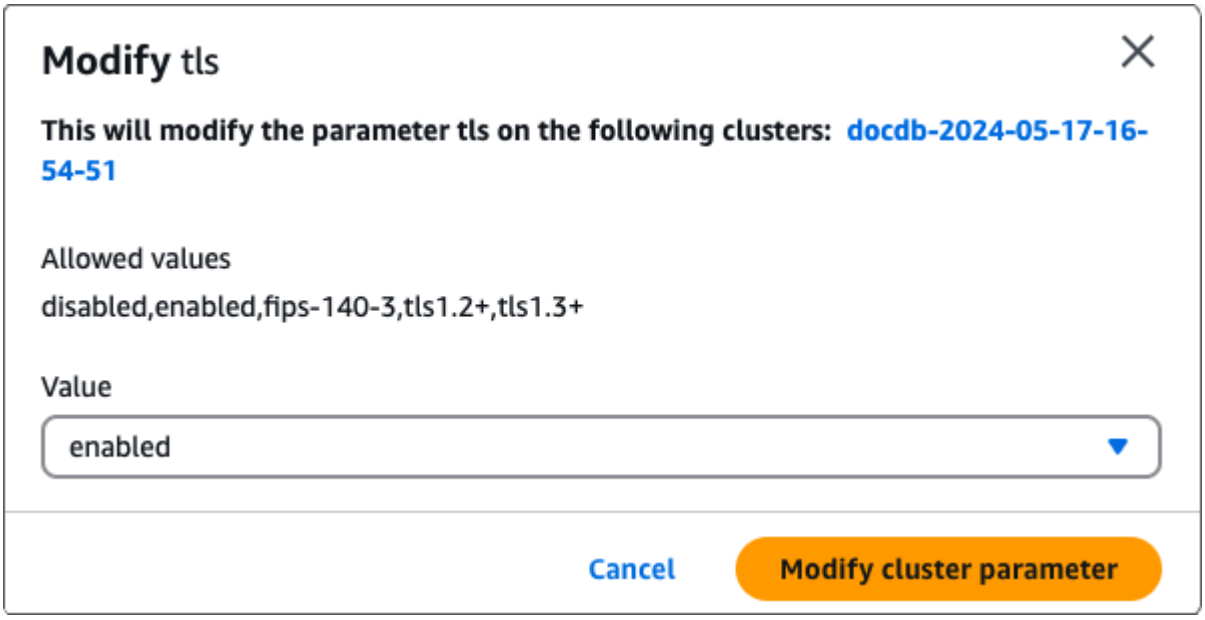
a. Elija el botón situado a la izquierda del nombre del parámetro de clúster (`tls`).

b. Elija Edit (Edición de).

c. Para cambiar el valor de `tls`, en el cuadro de diálogo Modificar**`tls`**, elija el valor que desee para el parámetro de clúster en el menú desplegable.

   Los valores válidos son los siguientes:

   - deshabilitado: deshabilita el TLS
   - habilitado: habilita TLS. Para Amazon DocumentDB 5.0 y versiones anteriores, habilita TLS 1.0 a 1.3, pero usa TLS 1.2 o superior. Amazon DocumentDB 5.0 solo admite TLS 1.2 y versiones posteriores, a partir de la versión secundaria 5.0.1, Amazon DocumentDB 8.0 y posteriores.
   - fips-140-3: habilita TLS con FIPS. El clúster solo acepta conexiones seguras según los requisitos de la publicación 140-3 de las Normas Federales de Proceso de la Información (FIPS). Esto solo se admite a partir de los clústeres de Amazon DocumentDB 5.0 (versión 3.0.3727 del motor) en estas regiones: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1.
   - tls1.2+: habilita las versiones 1.2 y superiores de TLS. Esto solo se admite a partir de Amazon DocumentDB 4.0 (versión de motor 2.0.10980) y Amazon DocumentDB (versión de motor 3.0.11051).
   - tls1.3+: habilita las versiones 1.3 y superiores de TLS. Esto solo se admite a partir de Amazon DocumentDB 4.0 (versión de motor 2.0.10980) y Amazon DocumentDB (versión de motor 3.0.11051).



d. Elija Modify cluster parameter (Modificar el parámetro de clúster). El cambio se aplicará a cada instancia del clúster cuando se reinicie.

4. Reinicie la instancia de Amazon DocumentDB.

   Reinicie todas las instancias del clúster para que el cambio se aplique a todas ellas.

   a. Abra la consola de https://console.aws.amazon.com/docdb Amazon DocumentDB en.

   b. En el panel de navegación, seleccione Instances (Instancias).

   c. Para especificar la instancia que se va a reiniciar, búsquela en la lista de instancias y elija el botón situado a la izquierda de su nombre.

   d. Elija Actions (Acciones) y, a continuación, Reboot (Reiniciar). Confirme que desea reiniciar eligiendo Reboot (Reiniciar).

Using the AWS CLI

Siga estos pasos para realizar tareas de administración para el cifrado TLS mediante la AWS CLI, como identificar grupos de parámetros, verificar el valor de TLS y realizar las modificaciones necesarias.

> **Note**
>
> A menos que especifique de manera diferente al crear un clúster, el clúster se crea con el grupo de parámetros de clúster predeterminado. Los parámetros del grupo de parámetros del `default` clúster no se pueden modificar (por ejemplo, `tls` enabled/disabled). Por lo tanto, si su clúster está utilizando un `default` grupo de parámetros de clúster, debe modificar el clúster para utilizar un grupo de parámetros de clúster no predeterminado. Es posible que primero tenga que crear un grupo de parámetros de clúster personalizado. Para obtener más información, consulte Creación de grupos de parámetros de clúster de Amazon DocumentDB.

1. Determine el grupo de parámetros de clúster que utiliza el clúster.

   Ejecute el comando `describe-db-clusters` con las siguientes opciones:

   - `--db-cluster-identifier`
   - `--query`

   En el siguiente ejemplo, reemplaza cada uno de *`user input placeholder`* ellos por la información de tu clúster.

   ```
   aws docdb describe-db-clusters \
     --db-cluster-identifier mydocdbcluster \
     --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
   ```

   La salida de esta operación será similar a lo que se indica a continuación (formato JSON):

   ```
   [
       [
           "mydocdbcluster",
           "myparametergroup"
       ]
   ]
   ```

   Si el nombre del grupo de parámetros de clúster es `default` (es decir, `default.docdb3.6`) debe tener un grupo de parámetros de clúster personalizado y

convertirlo en el grupo de parámetros de este clúster antes de continuar. Para obtener más información, consulte los temas siguientes:

1. Creación de grupos de parámetros de clúster de Amazon DocumentDB: si no dispone de un grupo de parámetros de clúster personalizado que pueda utilizar, créelo.
2. Modificación de un clúster de Amazon DocumentDB: modifique el clúster de forma que use el grupo de parámetros de clúster personalizado.

2. Determinar el valor actual del**`tls`** parámetro de clúster.

   Para obtener más información sobre este grupo de parámetros del clúster, ejecute el comando `describe-db-cluster-parameters` con las siguientes opciones:

   - `--db-cluster-parameter-group-name`
   - `--query`

     Limita la salida solo a los campos de interés: `ParameterName`, `ParameterValue`,`AllowedValues` y `ApplyType`.

   En el siguiente ejemplo, reemplaza cada una de *`user input placeholder`* ellas por la información de tu clúster.

```
aws docdb describe-db-cluster-parameters \
  --db-cluster-parameter-group-name myparametergroup \
  --query 'Parameters[*].[ParameterName,ParameterValue,AllowedValues,ApplyType]'
```

   La salida de esta operación será similar a lo que se indica a continuación (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. Modificar el valor del parámetro de clúster**`tls`**.

   Si el valor de `tls` no es el que necesita, modifíquelo para este grupo de parámetros de clúster. Para cambiar el valor del parámetro del clúster `tls`, utilice el comando `modify-db-cluster-parameter-group` con las siguientes opciones:

   - `--db-cluster-parameter-group-name`: obligatorio. El nombre del grupo de parámetros de clúster que se va a modificar. No puede ser el grupo de parámetros de clúster`default.*`.
   - `--parameters`: obligatorio. Lista de los parámetros del grupo de parámetros de clúster que se van a modificar.
     - `ParameterName`: obligatorio. El nombre del parámetro de clúster que se va a modificar.
     - `ParameterValue`: obligatorio. El valor nuevo de este parámetro de clúster. Debe ser uno de los `AllowedValues` del parámetro de clúster.
       - `enabled`— El clúster acepta conexiones seguras mediante TLS. Para Amazon DocumentDB 5.0 y versiones anteriores, esto habilita TLS 1.0 a 1.3, pero usa TLS 1.2 o superior. Amazon DocumentDB 5.0 solo admite TLS 1.2 y versiones posteriores, a partir de la versión secundaria 5.0.1, Amazon DocumentDB 8.0 y posteriores.
       - `disabled`: el clúster no acepta conexiones seguras mediante TLS.
       - `fips-140-3`: el clúster solo acepta conexiones seguras según los requisitos de la publicación 140-3 de las Normas Federales de Proceso de la Información (FIPS). Esto solo se admite a partir de los clústeres de Amazon DocumentDB 5.0 (versión 3.0.3727 del motor) en estas regiones: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1.
       - `tls1.2+`: el clúster acepta conexiones seguras mediante TLS, versión 1.2 y superiores. Esto solo se admite a partir de Amazon DocumentDB 4.0 (versión de motor 2.0.10980) y Amazon DocumentDB 5.0 (versión de motor 3.0.11051).

    - `tls1.3+`: el clúster acepta conexiones seguras mediante TLS, versión 1.3 y superiores. Esto solo se admite a partir de Amazon DocumentDB 4.0 (versión de motor 2.0.10980) y Amazon DocumentDB 5.0 (versión de motor 3.0.11051).
- `ApplyMethod`: cuándo se va a aplicar esta modificación. Para los parámetros de clúster estáticos, como `tle`, este valor debe ser `pending-reboot`.
    - `pending-reboot`: el cambio se aplica a una instancia solo después de reiniciarla. Debe reiniciar cada instancia de clúster por separado para que este cambio entre en vigor en todas las instancias del clúster.

En los siguientes ejemplos, sustituya cada uno por la información de su clúster. *`user input placeholder`*

El siguiente código deshabilita `tls`, y el cambio se aplica a cada instancia cuando se reinicia.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=tls,ParameterValue=disabled,ApplyMethod=pending-
reboot"
```

El siguiente código lo habilita `tls` y aplica el cambio a cada instancia cuando se reinicia.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=tls,ParameterValue=enabled,ApplyMethod=pending-
reboot"
```

El siguiente código habilita TLS con `fips-140-3`, y el cambio se aplica a cada instancia cuando se reinicia.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup2 \
  --parameters "ParameterName=tls,ParameterValue=fips-140-3,ApplyMethod=pending-
reboot"
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON):

```
{
    "DBClusterParameterGroupName": "myparametergroup"
```

```
}
```

4. Reinicie la instancia de Amazon DocumentDB.

   Reinicie todas las instancias del clúster para que el cambio se aplique a todas ellas. Para reiniciar una instancia de Amazon DocumentDB, utilice el comando `reboot-db-instance` con la siguiente opción:

   - `--db-instance-identifier`

   El siguiente código reinicia la instancia `mydocdbinstance`.

   En los siguientes ejemplos, sustituya cada uno por la *`user input placeholder`* información de su clúster.

   Example

   Para Linux, macOS o Unix:

```
aws docdb reboot-db-instance \
  --db-instance-identifier mydocdbinstance
```

   Para Windows:

```
aws docdb reboot-db-instance ^
  --db-instance-identifier mydocdbinstance
```

   La salida de esta operación será similar a lo que se indica a continuación (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"
    }
}
```

El reinicio de la instancia puede tardar unos minutos. Solo puede usar la instancia cuando su estado sea available (disponible). Puede monitorizar el estado de la instancia mediante la consola o la AWS CLI. Para obtener más información, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

Administración de claves

Amazon DocumentDB usa AWS Key Management Service (AWS KMS) para recuperar y administrar las claves de cifrado. AWS KMS combina hardware y software seguros y de alta disponibilidad

para proporcionar un sistema de administración de claves adaptado a la nube. Con AWS KMSél, puede crear claves de cifrado y definir las políticas que controlan la forma en que se pueden usar estas claves. AWS KMS admite AWS CloudTrail, por lo que puede auditar el uso de las claves para comprobar que las claves se utilizan correctamente.

Puede usar sus claves de KMS en combinación con Amazon DocumentDB y AWS los servicios compatibles, como Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), Amazon Elastic Block Store (Amazon EBS) y Amazon Redshift. Para obtener una lista de los servicios compatibles AWS KMS, consulte Cómo se usan AWS KMS los AWS servicios en la Guía para desarrolladores. AWS Key Management Service Para obtener más información al respecto AWS KMS, consulte ¿Qué es AWS Key Management Service?

Identity and Access Management para Amazon DocumentDB

AWS Identity and Access Management (IAM) ayuda al administrador Servicio de AWS a controlar de forma segura el acceso a AWS los recursos. Los administradores de IAM controlan quién se puede autenticar (iniciar sesión) y autorizar (tener permisos) para utilizar los recursos de Amazon DocumentDB. IAM es un Servicio de AWS servicio que puede utilizar sin coste adicional.

Temas

- Público
- Autenticación con identidades
- Administración del acceso con políticas
- Cómo funciona Amazon DocumentDB con IAM
- Identity-based ejemplos de políticas para Amazon DocumentDB
- Solución de problemas de identidad y acceso de Amazon DocumentDB
- Administración de permisos de acceso para los recursos de Amazon DocumentDB
- Uso de políticas basadas en identidad (políticas de IAM) para Amazon DocumentDB
- AWS políticas administradas para Amazon DocumentDB
- Permisos de la API de Amazon DocumentDB: referencia de acciones, recursos y condiciones

Público

La forma en que usas AWS Identity and Access Management (IAM) difiere según tu función:

- Usuario del servicio: solicite permisos al administrador si no puede acceder a las características (consulte Solución de problemas de identidad y acceso de Amazon DocumentDB).
- Administrador del servicio: determine el acceso de los usuarios y envíe las solicitudes de permiso (consulte Cómo funciona Amazon DocumentDB con IAM).
- Administrador de IAM: escribe las políticas para administrar el acceso (consulte Identity-based ejemplos de políticas para Amazon DocumentDB).

Autenticación con identidades

La autenticación es la forma en que inicias sesión AWS con tus credenciales de identidad. Debe autenticarse como usuario de Usuario raíz de la cuenta de AWS IAM o asumiendo una función de IAM.

Puedes iniciar sesión como una identidad federada con credenciales de una fuente de identidad, como AWS IAM Identity Center (el centro de identidad de IAM), la autenticación de inicio de sesión único o las credenciales. Google/Facebook Para obtener más información sobre el inicio de sesión, consulte Cómo iniciar sesión en la Cuenta de AWS en la Guía del usuario de AWS Sign-In .

Para el acceso programático, AWS proporciona un SDK y una CLI para firmar criptográficamente las solicitudes. Para obtener más información, consulte AWS Signature Version 4 para solicitudes de API en la Guía del usuario de IAM.

Cuenta de AWS usuario raíz

Cuando creas una Cuenta de AWS, comienzas con una identidad de inicio de sesión denominada usuario Cuenta de AWS raíz que tiene acceso total a todos los recursos. Servicios de AWS Se recomiendaencarecidamente que no utilice el usuario raíz para las tareas diarias. Para ver las tareas que requieren credenciales de usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos que utilicen la federación con un proveedor de identidad para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario de su directorio empresarial, proveedor de identidad web o Directory Service que accede Servicios de AWS mediante credenciales de una fuente de identidad. Las identidades federadas asumen roles que proporcionan credenciales temporales.

Para una administración de acceso centralizada, se recomienda AWS IAM Identity Center. Para obtener más información, consulte ¿Qué es el Centro de identidades de IAM? en la Guía del usuario de AWS IAM Identity Center .

Usuarios y grupos de IAM

Un usuario de IAM es una identidad con permisos específicos para una sola persona o aplicación. Recomendamos el uso de credenciales temporales en lugar de usuarios de IAM con credenciales de larga duración. Para obtener más información, consulte Exigir a los usuarios humanos que utilicen la federación con un proveedor de identidad para acceder AWS mediante credenciales temporales en la Guía del usuario de IAM.

Un grupo de IAM especifica un conjunto de usuarios de IAM y facilita la administración de los permisos para grupos grandes de usuarios. Para obtener más información, consulte Casos de uso para usuarios de IAM en la Guía del usuario de IAM.

Roles de IAM

Un Rol de IAM es una identidad con permisos específicos que proporciona credenciales temporales. Puede asumir un rol cambiando de un rol de usuario a uno de IAM (consola) o llamando a una AWS CLI operación de AWS API. Para obtener más información, consulte Métodos para asumir un rol en la Guía del usuario de IAM.

Los roles de IAM son útiles para el acceso de usuario federado, los permisos de usuario de IAM temporales, el acceso entre cuentas, el acceso entre servicios y las aplicaciones que se ejecutan en Amazon EC2. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Administración del acceso con políticas

Puedes controlar el acceso AWS creando políticas y adjuntándolas a las AWS identidades o los recursos. Una política define los permisos cuando se asocian a una identidad o un recurso. AWS evalúa estas políticas cuando un director hace una solicitud. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre los documentos de políticas de JSON, consulte Información general de políticas de JSON en la Guía del usuario de IAM.

Mediante las políticas, los administradores especifican quién tiene acceso a qué, definiendo qué entidad principal puede realizar acciones sobre qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM crea políticas de IAM y las agrega a roles, que los usuarios pueden asumir posteriormente. Las políticas de IAM definen permisos independientemente del método que se utilice para realizar la operación.

Identity-based políticas

Identity-based las políticas son documentos de política de permisos de JSON que se adjuntan a una identidad (usuario, grupo o rol). Estas políticas controlan qué acciones pueden realizar las identidades, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en la identidad, consulte Definición de permisos de IAM personalizados con políticas administradas por el cliente en la Guía del usuario de IAM.

Identity-based las políticas pueden ser políticas en línea (integradas directamente en una sola identidad) o políticas administradas (políticas independientes asociadas a varias identidades). Para obtener información sobre cómo elegir entre políticas administradas e insertadas, consulte Selección entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Resource-based políticas

Resource-based las políticas son documentos de políticas JSON que se adjuntan a un recurso. Los ejemplos incluyen las Políticas de confianza de roles de IAM y las Políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Debe especificar una entidad principal en una política basada en recursos.

Resource-based las políticas son políticas en línea que se encuentran en ese servicio. No puede usar políticas AWS administradas de IAM en una política basada en recursos.

Otros tipos de políticas

AWS admite tipos de políticas adicionales que pueden establecer los permisos máximos que otorgan los tipos de políticas más comunes:

- Límites de permisos: establecen los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM. Para obtener más información, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCP): especifican los permisos máximos para una organización o unidad organizativa en AWS Organizations. Para obtener más información, consulte Políticas de control de servicios en la Guía del usuario de AWS Organizations .

- Políticas de control de recursos (RCP): definen los permisos máximos disponibles para los recursos de las cuentas. Para obtener más información, consulte Políticas de control de recursos (RCP) en la Guía del usuario de AWS Organizations .
- Políticas de sesión: políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal para un rol o un usuario federado. Para obtener más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo se AWS determina si se permite una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la guía del usuario de IAM.

Cómo funciona Amazon DocumentDB con IAM

Antes de utilizar IAM para administrar el acceso a Amazon DocumentDB, obtenga información sobre qué características de IAM se encuentran disponibles con Amazon DocumentDB.

Características de IAM que puede utilizar con Amazon DocumentDB

Característica de IAM	Instance-based clústeres	Clústeres elásticos
Identity-based políticas	Sí	Sí
Resource-based políticas	No	No
Acciones de políticas	Sí	Sí
Recursos de políticas	Sí	Sí
Claves de condición de política (específicas del servicio)	Sí	Sí
ACL	No	No
ABAC (etiquetas en políticas)	Parcial	Sí

Característica de IAM	Instance-based clústeres	Clústeres elásticos
Credenciales temporales	Sí	Sí
Permisos de entidades principales	Sí	Sí
Roles de servicio	Sí	Sí
Service-linked roles	No	Sí

Para obtener una visión general de cómo Amazon DocumentDB y otros AWS servicios funcionan con la mayoría de las funciones de IAM, consulte AWS los servicios que funcionan con IAM en la Guía del usuario de IAM.

Identity-based políticas de Amazon DocumentDB

Compatibilidad con las políticas basadas en identidad: sí

Identity-based las políticas son documentos de política de permisos de JSON que puedes adjuntar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en la identidad, consulte Definición de permisos de IAM personalizados con políticas administradas por el cliente en la Guía del usuario de IAM.

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. Para obtener más información sobre los elementos que puede utilizar en una política de JSON, consulte Referencia de los elementos de la política de JSON de IAM en la Guía del usuario de IAM.

Identity-based ejemplos de políticas para Amazon DocumentDB

Para ver ejemplos de políticas basadas en identidad de Amazon DocumentDB, consulte Identity-based ejemplos de políticas para Amazon DocumentDB.

Resource-based políticas dentro de Amazon DocumentDB

Admite políticas basadas en recursos: no

Resource-based las políticas son documentos de políticas JSON que se adjuntan a un recurso. Los ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política basada en recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Para habilitar el acceso entre cuentas, puede especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política en función de recursos. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Acciones de políticas para Amazon DocumentDB

Compatibilidad con las acciones de políticas: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

> **Note**
>
> Para determinadas funciones de administración, Amazon DocumentDB utiliza tecnología operativa que se comparte con Amazon Relational Database Service (Amazon RDS). Para ver una lista de las acciones de Amazon DocumentDB, consulte Acciones definidas por el servicio de base de datos relacional de Amazon en la Referencia de autorizaciones de servicio.
> Para ver las acciones de políticas para los clústeres elásticos de Amazon DocumentDB, consulte Acciones definidas por los clústeres elásticos de Amazon DocumentDB en la Referencia de autorización de servicios.

Las acciones de políticas de Amazon DocumentDB utilizan el siguiente prefijo antes de la acción:

```
aws
```

Para especificar varias acciones en una única instrucción, sepárelas con comas.

```
"Action": [
      "aws:action1",
      "aws:action2"
         ]
```

Para ver ejemplos de políticas basadas en identidad de Amazon DocumentDB, consulte Identity-based ejemplos de políticas para Amazon DocumentDB.

Recursos de políticas para Amazon DocumentDB

Compatibilidad con los recursos de políticas: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). En el caso de las acciones que no admiten permisos por recurso, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

> **Note**
>
> Para determinadas funciones de administración, Amazon DocumentDB utiliza tecnología operativa que se comparte con Amazon Relational Database Service (Amazon RDS). Para ver una lista de tipos de recursos y sus ARN, consulte Tipos de recurso definidos por el servicio de base de datos relacional de Amazon en la Referencia de autorizaciones de servicio. Para obtener información acerca de las acciones con las que puede especificar el ARN de cada recurso, consulte Acciones definidas por el servicio de base de datos relacional de Amazon.

Para ver los tipos de recursos de los clústeres elásticos de Amazon DocumentDB, consulte los tipos de recursos definidos por los clústeres elásticos de Amazon DocumentDB en la referencia de autorización de servicios.

Para ver ejemplos de políticas basadas en identidad de Amazon DocumentDB, consulte Identity-based ejemplos de políticas para Amazon DocumentDB.

Claves de condición de políticas para Amazon DocumentDB

Compatibilidad con claves de condición de políticas específicas del servicio: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` especifica cuándo se ejecutan las instrucciones en función de criterios definidos. Puede crear expresiones condicionales que utilizan operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud. Para ver todas las claves de condiciones AWS globales, consulte las claves de contexto de condiciones AWS globales en la guía del usuario de IAM.

> **Note**
>
> Para determinadas funciones de administración, Amazon DocumentDB utiliza tecnología operativa que se comparte con Amazon Relational Database Service (Amazon RDS). Para obtener una lista de las claves de condición de Amazon ECS, consulte Claves de condición del servicio de base de datos relacional de Amazon en la Referencia de autorizaciones de servicio. Para obtener más información acerca de las acciones y los recursos con los que puede utilizar una clave de condición, consulte Acciones definidas por el servicio de base de datos relacional de Amazon.
> Para ver las claves de condición de los clústeres elásticos de Amazon DocumentDB, consulte las claves de condición de los clústeres elásticos de Amazon DocumentDB en la Referencia de autorización de servicios.

Para ver ejemplos de políticas basadas en identidad de Amazon DocumentDB, consulte Identity-based ejemplos de políticas para Amazon DocumentDB.

ACL en Amazon DocumentDB

Compatibilidad con ACL: no

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

ABAC con Amazon DocumentDB

> **Note**
>
> ABAC solo es compatible parcialmente con los clústeres basados en instancias, pero es compatible con los clústeres elásticos.

Attribute-based El control de acceso (ABAC) es una estrategia de autorización que define los permisos en función de atributos denominados etiquetas. Puede adjuntar etiquetas a las entidades y AWS los recursos de IAM y, a continuación, diseñar políticas de ABAC para permitir las operaciones cuando la etiqueta del principal coincida con la etiqueta del recurso.

Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Si un servicio admite las tres claves de condición para cada tipo de recurso, el valor es Sí para el servicio. Si un servicio admite las tres claves de condición solo para algunos tipos de recursos, el valor es Parcial.

Para obtener más información sobre ABAC, consulte Definición de permisos con la autorización de ABAC en la Guía del usuario de IAM. Para ver un tutorial con los pasos para configurar ABAC, consulte Uso del control de acceso basado en atributos (ABAC) en la Guía del usuario de IAM.

Uso de credenciales temporales con Amazon DocumentDB

Compatibilidad con credenciales temporales: sí

Las credenciales temporales proporcionan acceso a corto plazo a AWS los recursos y se crean automáticamente al usar la federación o al cambiar de rol. AWS recomienda generar credenciales temporales de forma dinámica en lugar de utilizar claves de acceso a largo plazo. Para obtener

más información, consulte Credenciales de seguridad temporales en IAM y Servicios de AWS que funcionan con IAM en la Guía del usuario de IAM.

Cross-service permisos principales para Amazon DocumentDB

Admite sesiones de acceso directo (FAS): sí

Las sesiones de acceso directo (FAS) utilizan los permisos de la persona principal que llama a una Servicio de AWS, combinados con la solicitud Servicio de AWS para realizar solicitudes a los servicios descendentes. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Sesiones de acceso directo.

Roles de servicio para Amazon DocumentDB

Compatible con roles de servicio: sí

Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Crear un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

> **Warning**
>
> Cambiar los permisos de un rol de servicio podría interrumpir la funcionalidad de Amazon DocumentDB. Edite los roles de servicio solo cuando Amazon DocumentDB proporcione orientación para hacerlo.

Service-linked roles de Amazon DocumentDB

> **Note**
>
> Service-linked los roles no se admiten para los clústeres basados en instancias, pero sí para los clústeres elásticos.

Un rol vinculado a un servicio es un tipo de rol de servicio que está vinculado a un. Servicio de AWS El servicio puede asumir la función de realizar una acción en su nombre. Service-linked las funciones aparecen en su nombre Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

Para más información sobre cómo crear o administrar roles vinculados a servicios, consulta Servicios de AWS que funcionan con IAM. Busque un servicio en la tabla que incluya un servicio `Yes` en la columna de Service-linked funciones. Seleccione el vínculo Sí para ver la documentación acerca del rol vinculado a servicios para ese servicio.

Identity-based ejemplos de políticas para Amazon DocumentDB

De forma predeterminada, los usuarios y roles no tienen permiso para crear ni modificar los recursos de Amazon DocumentDB. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan.

Para obtener información acerca de cómo crear una política basada en identidades de IAM mediante el uso de estos documentos de políticas JSON de ejemplo, consulte Creación de políticas de IAM (consola) en la Guía del usuario de IAM.

A fin de obtener más información sobre las acciones y los tipos de recursos definidos por Amazon DocumentDB, incluido el formato de los ARN para cada tipo de recurso, consulte Acciones, recursos y claves para el servicio de base de datos relacional de Amazon en la Referencia de autorizaciones de servicio.

> **Note**
>
> Para determinadas funciones de administración, Amazon DocumentDB utiliza tecnología operativa que se comparte con Amazon Relational Database Service (Amazon RDS).
> Para conocer las acciones de políticas para los clústeres elásticos de Amazon DocumentDB, consulte Acciones, recursos y claves de condición de los clústeres elásticos de Amazon DocumentDB en la Referencia de autorización de servicios.

Temas

- Prácticas recomendadas sobre las políticas
- Uso de la consola de Amazon DocumentDB
- Cómo permitir a los usuarios consultar sus propios permisos

Prácticas recomendadas sobre las políticas

Identity-based las políticas determinan si alguien puede crear recursos de Amazon DocumentDB de su cuenta, acceder a ellos o eliminarlos. Estas acciones pueden generar costos adicionales para

su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas AWS administradas que otorgan permisos en muchos casos de uso comunes. Están disponibles en su. Cuenta de AWS Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por los AWS clientes que sean específicas para sus casos de uso. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de tarea en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder acceso a las acciones de servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.
- Utiliza el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Validación de políticas con el Analizador de acceso de IAM en la Guía del usuario de IAM.
- Exigir la autenticación multifactor (MFA): si tienes una situación en la que se requieren usuarios de IAM o un usuario raíz Cuenta de AWS, activa la MFA para aumentar la seguridad. Para exigir la MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA a sus políticas. Para más información, consulte Acceso seguro a la API con MFA en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Uso de la consola de Amazon DocumentDB

Para acceder a la consola de Amazon DocumentDB (compatible con MongoDB) , debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y ver detalles sobre los recursos de Amazon DocumentDB que tiene en su cuenta. Cuenta de AWS Si crea una política basada en identidades que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

No es necesario conceder permisos mínimos de consola a los usuarios que solo realizan llamadas a la API AWS CLI o a la AWS API. En su lugar, permita el acceso únicamente a las acciones que coincidan con la operación de API que intentan realizar.

Para garantizar que los usuarios y los roles puedan seguir utilizando la consola de Amazon DocumentDB, adjunte también la política *ReadOnly* AWS gestionada *ConsoleAccess* o de Amazon DocumentDB a las entidades. Para obtener más información, consulte Adición de permisos a un usuario en la Guía del usuario de IAM:

Cómo permitir a los usuarios consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante la 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": "*"
        }
    ]
}
```

Solución de problemas de identidad y acceso de Amazon DocumentDB

Utilice la siguiente información para diagnosticar y solucionar los problemas comunes que es posible que surjan cuando se trabaja con Amazon DocumentDB e IAM.

Temas

- No tengo autorización para realizar una acción en Amazon DocumentDB
- No estoy autorizado para realizar lo siguiente: PassRole
- Quiero permitir la entrada a personas ajenas a mi Cuenta de AWS para acceder a mis recursos de Amazon DocumentDB

No tengo autorización para realizar una acción en Amazon DocumentDB

Si recibe un error que indica que no tiene autorización para realizar una acción, las políticas se deben actualizar para permitirle realizar la acción.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM `mateojackson` intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *`my-example-widget`*, pero no tiene los permisos ficticios `aws:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 aws:GetWidget on resource: my-example-widget
```

En este caso, la política del usuario `mateojackson` debe actualizarse para permitir el acceso al recurso *`my-example-widget`* mediante la acción `aws:GetWidget`.

Si necesitas ayuda, ponte en contacto con tu administrador. AWS El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

No estoy autorizado para realizar lo siguiente: PassRole

Si recibe un error que indica que no tiene autorización para llevar a cabo la acción `iam:PassRole`, las políticas se deben actualizar para permitirle pasar un rol a Amazon DocumentDB.

Algunas Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada a un servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en Amazon DocumentDB. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su AWS administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir la entrada a personas ajenas a mi Cuenta de AWS para acceder a mis recursos de Amazon DocumentDB

Se puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Se puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para obtener más información, consulte lo siguiente:

- Para saber si Amazon DocumentDB admite estas características, consulte Cómo funciona Amazon DocumentDB con IAM.
- Para obtener información sobre cómo proporcionar acceso a los recursos de su propiedad Cuentas de AWS , consulte Cómo proporcionar acceso a un usuario de IAM en otro Cuenta de AWS que sea de su propiedad en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo proporcionar acceso a los recursos que Cuentas de AWS pertenecen a terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulte Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para conocer sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Administración de permisos de acceso para los recursos de Amazon DocumentDB

Cada AWS recurso es propiedad de un Cuenta de AWS, y los permisos para crear o acceder a los recursos se rigen por las políticas de permisos. Un administrador de cuentas puede adjuntar políticas de permisos a las identidades de IAM (es decir, usuarios, grupos y roles), y algunos servicios (por ejemplo AWS Lambda) también admiten adjuntar políticas de permisos a los recursos.

> **Note**
>
> Un administrador de la cuenta (o usuario administrador) es un usuario con permisos de administrador. Para obtener más información, consulte Prácticas recomendadas de IAM en la Guía del usuario de IAM.

Temas

- Recursos y operaciones de Amazon DocumentDB
- Titularidad de los recursos
- Administración del acceso a los recursos
- Especificación de elementos de política: acciones, efectos, recursos y entidades principales

- Especificación de las condiciones de una política

Recursos y operaciones de Amazon DocumentDB

En Amazon DocumentDB, el recurso principal es un clúster. Amazon DocumentDB admite otros recursos que pueden utilizarse con el recurso principal, como las instancias, los grupos de parámetros y las suscripciones a eventos. Estos recursos se denominan subrecursos.

Estos recursos principales y secundarios tienen asociado un Nombre de recursos de Amazon (ARN) único, tal y como se muestra en la siguiente tabla:

Tipo de recurso	Formato de ARN
Clúster	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster:` *`db-cluster-name`*
Grupo de parámetros del clúster	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster-pg:` *`cluster-parameter-group-name`*
Instantánea de clúster	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster-snapshot:` *`cluster-snapshot-name`*
Instancia	`arn:aws:rds:` *`region`*`:`*`account-id`* `:db:`*`db-instance-name`*
Security group (Grupo de seguridad)	`arn:aws:rds:` *`region`*`:`*`account-id`* `:secgrp:`*`security-group-name`*
Grupo de subredes	`arn:aws:rds:` *`region`*`:`*`account-id`* `:subgrp:`*`subnet-group-name`*

Amazon DocumentDB proporciona un conjunto de operaciones para trabajar con los recursos de Amazon DocumentDB. Para obtener una lista de operaciones disponibles, consulte Acciones.

Titularidad de los recursos

El propietario de un recurso es el Cuenta de AWS que creó un recurso. Es decir, el propietario del recurso es el Cuenta de AWS de la entidad principal (la cuenta raíz, un usuario de IAM o un rol de IAM) que autentica la solicitud que crea el recurso. Los siguientes ejemplos ilustran cómo funciona:

- Si utiliza las credenciales de su cuenta raíz Cuenta de AWS para crear un recurso de Amazon DocumentDB, como una instancia, Cuenta de AWS es el propietario del recurso de Amazon DocumentDB.
- Si crea un usuario de IAM en su cuenta Cuenta de AWS y le concede permisos para crear recursos de Amazon DocumentDB, el usuario puede crear recursos de Amazon DocumentDB. Sin embargo, usted Cuenta de AWS, al que pertenece el usuario, es el propietario de los recursos de Amazon DocumentDB.
- Si crea un rol de IAM Cuenta de AWS con permisos para crear recursos de Amazon DocumentDB, cualquier persona que pueda asumir el rol puede crear recursos de Amazon DocumentDB. Usted Cuenta de AWS, al que pertenece el rol, es el propietario de los recursos de Amazon DocumentDB.

Administración del acceso a los recursos

Una política de permisos describe quién tiene acceso a qué. En la siguiente sección se explican las opciones disponibles para crear políticas de permisos.

> **Note**
>
> En esta sección, también se explica el uso de IAM en el contexto de Amazon DocumentDB. No se proporciona información detallada sobre el servicio de IAM. Para ver la documentación completa de IAM, consulte ¿Qué es IAM? en la Guía del usuario de IAM. Para obtener más información acerca de la sintaxis y las descripciones de las políticas de IAM, consulte Referencia de políticas de IAM de AWS en la Guía del usuario de IAM.

Las políticas que se asocian a una identidad de IAM se denominan políticas basadas en la identidad (políticas de IAM). Las políticas que se asocian a un recurso se denominan "políticas basadas en recursos". Amazon DocumentDB solo admite políticas basadas en identidad (políticas del IAM).

Temas

- Identity-based políticas (políticas de IAM)
- Resource-based políticas

Identity-based políticas (políticas de IAM)

Puede asociar políticas a identidades de IAM. Por ejemplo, puede hacer lo siguiente:

- Asociar una política de permisos a un usuario o un grupo de su cuenta: un administrador de la cuenta puede utilizar una política de permisos asociada a un usuario determinado para concederle permisos para crear un recurso de Amazon DocumentDB, como una instancia.
- Adjuntar una política de permisos a un rol (conceder permisos para cuentas cruzadas): puede adjuntar una política de permisos basada en identidades a un rol de IAM para conceder permisos para cuentas cruzadas. Por ejemplo, un administrador puede crear una función para conceder permisos multicuenta a otra Cuenta de AWS o a un AWS servicio de la siguiente manera:
  1. El administrador de la CuentaA crea un rol de IAM y asocia una política de permisos a dicho rol, que concede permisos sobre los recursos de la CuentaA.
  2. El administrador de la CuentaA asocia una política de confianza al rol que identifica la Cuenta B como la entidad principal que puede asumir el rol.
  3. El administrador de la cuenta B puede entonces delegar los permisos para asumir la función a cualquier usuario de la cuenta B. De este modo, los usuarios de la cuenta B pueden crear o acceder a los recursos de la cuenta A. El director de la política de confianza también puede ser un director de AWS servicio si desea conceder permisos a un AWS servicio para que asuma el rol.

  Para obtener más información sobre el uso de IAM para delegar permisos, consulte Administración de accesos en la Guía del usuario de IAM.

A continuación se muestra un ejemplo de una política que permite al usuario con el ID `123456789012` crear instancias para una Cuenta de AWS. La nueva instancia debe usar un grupo de opciones y un grupo de parámetros de base de datos que comience por `default` y debe utilizar el grupo de subredes `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"
            ]
        }
    ]
}
```

Para obtener más información acerca del uso de políticas basadas en identidades con Amazon DocumentDB, consulte Uso de políticas basadas en identidad (políticas de IAM) para Amazon DocumentDB. Para obtener más información sobre usuarios, grupos, roles y permisos, consulte Identidades (usuarios, grupos y roles) en la Guía del usuario de IAM.

Resource-based políticas

Otros servicios de, como Amazon Simple Storage Service (Amazon S3), también admiten políticas de permisos basadas en recursos. Por ejemplo, puede adjuntar una política a un bucket de Amazon S3 para administrar los permisos de acceso a dicho bucket. Amazon DocumentDB no admite políticas basadas en recursos.

Especificación de elementos de política: acciones, efectos, recursos y entidades principales

Para cada recurso de Amazon DocumentDB (consulte Recursos y operaciones de Amazon DocumentDB), el servicio define un conjunto de operaciones de API. Para obtener más información, consulte Acciones. Para conceder permisos para estas operaciones de la API, Amazon DocumentDB define un conjunto de acciones que usted puede especificar en una política. Para realizar una operación API pueden ser necesarios permisos para más de una acción.

A continuación, se indican los elementos básicos de la política:

- Recurso: en una política, se usa un nombre de recurso de Amazon (ARN) para identificar el recurso al que se aplica la política.
- Acción: utilice palabras clave de acción para identificar las operaciones del recurso que desea permitir o denegar. Por ejemplo, cuando se concede el permiso `rds:DescribeDBInstances`, el usuario puede realizar la operación `DescribeDBInstances`.

- Efecto: especifique el efecto que se producirá cuando el usuario solicite la acción específica; puede ser permitir o denegar. Si no concede acceso de forma explícita (permitir) a un recurso, el acceso se deniega implícitamente. También puede denegar explícitamente el acceso a un recurso para asegurarse de que un usuario no pueda obtener acceso a él, aunque otra política le conceda acceso.
- Entidad principal: en las políticas basadas en identidades (políticas de IAM), el usuario al que se asocia esta política es la entidad principal implícita. Para las políticas basadas en recursos, debe especificar el usuario, la cuenta, el servicio u otra entidad que desee que reciba permisos (se aplica solo a las políticas basadas en recursos). Amazon DocumentDB no admite políticas basadas en recursos.

Para obtener más información sobre la sintaxis y descripciones de las políticas de IAM, consulte Referencia de la política de IAM de AWS en la Guía del usuario de IAM.

Para ver una tabla con todas las acciones de la API de Amazon DocumentDB y los recursos a los que se aplican, consulte Permisos de la API de Amazon DocumentDB: referencia de acciones, recursos y condiciones.

Especificación de las condiciones de una política

Al conceder permisos, puede utilizar el lenguaje de la política de IAM para especificar las condiciones en la que se debe aplicar una política. Por ejemplo, es posible que desee que solo se aplique una política después de una fecha específica. Para obtener más información sobre cómo especificar condiciones en un lenguaje de política, consulte Condition en la Guía del usuario de IAM.

Para expresar condiciones, se usan claves de condición predefinidas. Amazon DocumentDB no tiene claves de contexto específicas de servicios que se puedan utilizar en una política de IAM. Para obtener una lista de las claves de contexto de condiciones globales que están disponibles para todos los servicios, consulte las claves de condiciones globales disponibles en la Guía del usuario de IAM.

Uso de políticas basadas en identidad (políticas de IAM) para Amazon DocumentDB

> **Important**
>
> En determinadas características de administración, Amazon DocumentDB utiliza una tecnología operativa que comparte con Amazon RDS. Las llamadas a la consola y a la API

de Amazon DocumentDB se registran como llamadas realizadas a la API de Amazon RDS.
AWS CLI
En primer lugar, revise los temas introductorios que explican los conceptos básicos y las opciones para administrar el acceso a los recursos de Amazon DocumentDB. Para obtener más información, consulte Administración de permisos de acceso para los recursos de Amazon DocumentDB.

Este tema contiene ejemplos de políticas basadas en identidades, donde los administradores de cuentas pueden asociar políticas de permisos a identidades de IAM (es decir, a usuarios, grupos y funciones).

A continuación, se muestra un ejemplo de política de 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"
            ]
        }
    ]
}
```

En la política se incluye una sola instrucción que especifica los siguientes permisos para el usuario de IAM:

- La política permite al usuario de IAM crear una instancia mediante la acción CreateDBInstance (esto también se aplica a la operación https://docs.aws.amazon.com/cli/latest/reference/rds/create-db-instance.html AWS CLI create-db-instance y a la). Consola de administración de AWS
- El elemento `Resource` especifica que el usuario puede realizar acciones en o con recursos. Puede especificar los recursos mediante un nombre de recurso de Amazon (ARN). Este ARN incluye el nombre del servicio al que pertenece el recurso (), el Región de AWS (*indica cualquier región en este ejemplo`rds`), el número de cuenta de usuario (`123456789012`es el identificador de usuario en este ejemplo) y el tipo de recurso.

  El elemento `Resource` del ejemplo especifica las siguientes restricciones políticas en los recursos del usuario:

  - El identificador de instancia para la nueva instancia de base de datos debe comenzar por `test` (por ejemplo, `testCustomerData1`, `test-region2-data`).
  - El grupo de parámetros de clúster de la nueva instancia debe empezar por `default`.
  - El grupo de subredes de la nueva instancia debe ser el grupo de subredes `default`.

La política no especifica el elemento `Principal`, ya que en una política basada en la identidad no se especifica la entidad principal que obtiene el permiso. Al asociar una política a un usuario, el usuario es la entidad principal implícita. Cuando asocia una política de permisos a un rol de IAM, la entidad principal identificada en la política de confianza de rol obtiene los permisos.

Para ver una tabla con todas las operaciones de la API de Amazon DocumentDB y los recursos a los que se aplican, consulte Permisos de la API de Amazon DocumentDB: referencia de acciones, recursos y condiciones.

Permisos necesarios para usar la consola de Amazon DocumentDB

Para que un usuario pueda trabajar con la consola Amazon DocumentDB, debe tener un conjunto mínimo de permisos. Estos permisos permiten al usuario describir los recursos de Amazon DocumentDB que tiene Cuenta de AWS y proporcionar otra información relacionada, incluida la información de red y seguridad de Amazon EC2.

Si crea una política de IAM que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para los usuarios con esa política de IAM. Para asegurarse de que esos usuarios puedan seguir usando la consola Amazon DocumentDB, asocie también la política administrada `AmazonDocDBConsoleFullAccess` al usuario, según se explica en AWS políticas administradas para Amazon DocumentDB.

No es necesario conceder permisos mínimos de consola a los usuarios que solo realizan llamadas a la API de Amazon DocumentDB AWS CLI o a la API de Amazon DocumentDB.

Ejemplos de políticas administradas por el cliente

En esta sección, encontrará ejemplos de políticas de usuario que conceden permisos para diversas acciones de Amazon DocumentDB. Estas políticas funcionan cuando utiliza las acciones de la API, los SDK o los AWS SDK de Amazon DocumentDB. AWS CLI Cuando se utiliza la consola, debe conceder permisos adicionales específicos a la consola, tal y como se explica en Permisos necesarios para usar la consola de Amazon DocumentDB.

Para determinadas funciones de administración, Amazon DocumentDB utiliza tecnología operativa que se comparte con Amazon Relational Database Service (Amazon RDS) y Amazon Neptune.

> **Note**
>
> Todos los ejemplos utilizan la región Este de EE. UU. (Norte de Virginia) (`us-east-1`) y contienen identificadores de cuenta ficticios.

Ejemplos

- Ejemplo 1: permitir que un usuario lleve a cabo cualquier acción Describe con cualquier recurso de Amazon DocumentDB
- Ejemplo 2: prevención de la eliminación de una instancia por parte de un usuario
- Ejemplo 3: prevención de la creación de un clúster por parte de un usuario a menos que el cifrado de almacenamiento esté habilitado

Ejemplo 1: permitir que un usuario lleve a cabo cualquier acción Describe con cualquier recurso de Amazon DocumentDB

La siguiente política de permisos concede permisos a un usuario para ejecutar todas las acciones que empiezan por `Describe`. Estas acciones muestran información acerca de un recurso de Amazon DocumentDB, como una instancia. El carácter comodín (*) del elemento `Resource` indica que las acciones están permitidas en todos los recursos de Amazon DocumentDB que son propiedad de la cuenta.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowRDSDescribe",
         "Effect":"Allow",
         "Action":"rds:Describe*",
         "Resource":"*"
      }
   ]
}
```

Ejemplo 2: prevención de la eliminación de una instancia por parte de un usuario

La siguiente política de permisos concede permisos para impedir que un usuario elimine una instancia específica. Por ejemplo, puede servir para impedir la eliminación de instancias de producción a cualquier usuario que no sea un administrador.

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"
      }
   ]
}
```

Ejemplo 3: prevención de la creación de un clúster por parte de un usuario a menos que el cifrado de almacenamiento esté habilitado

La siguiente política de permisos deniega los permisos a un usuario para crear un clúster de Amazon DocumentDB a menos que se habilite el cifrado de almacenamiento.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "PreventUnencryptedDocumentDB",
         "Effect": "Deny",
         "Action": "RDS:CreateDBCluster",
         "Condition": {
         "Bool": {
         "rds:StorageEncrypted": "false"
      },
         "StringEquals": {
         "rds:DatabaseEngine": "docdb"
         }
      },
      "Resource": "*"
      }
   ]
}
```

AWS políticas administradas para Amazon DocumentDB

Para añadir permisos a los usuarios, grupos y roles, es más fácil usar políticas AWS administradas que escribirlas usted mismo. Se necesita tiempo y experiencia para crear políticas administradas por el cliente de IAM que proporcionen a su equipo solo los permisos necesarios. Para empezar rápidamente, puedes usar nuestras políticas AWS administradas. Estas políticas cubren casos de uso comunes y están disponibles en su Cuenta de AWS. Para obtener más información sobre las políticas AWS administradas, consulte las políticas AWS administradas en la Guía del usuario de IAM.

AWS los servicios mantienen y AWS actualizan las políticas administradas. No puede cambiar los permisos de las políticas AWS administradas. En ocasiones, los servicios agregan permisos adicionales a una política AWS administrada para admitir nuevas funciones. Este tipo de actualización afecta a todas las identidades (usuarios, grupos y roles) donde se asocia la política. Es más probable que los servicios actualicen una política AWS administrada cuando se lanza una nueva función o cuando hay nuevas operaciones disponibles. Los servicios no eliminan los permisos de una

política AWS administrada, por lo que las actualizaciones de las políticas no anularán los permisos existentes.

Además, AWS admite políticas administradas para funciones laborales que abarcan varios servicios. Por ejemplo, la política `ViewOnlyAccess` AWS administrada proporciona acceso de solo lectura a muchos AWS servicios y recursos. Cuando un servicio lanza una nueva función, AWS agrega permisos de solo lectura para nuevas operaciones y recursos. Para obtener una lista y descripciones de las políticas de funciones de trabajo, consulte Políticas administradas de AWS para funciones de trabajo en la Guía del usuario de IAM.

Las siguientes políticas AWS administradas, que puede adjuntar a los usuarios de su cuenta, son específicas de Amazon DocumentDB:

- AmazonDocDBFullAccess— Otorga acceso total a todos los recursos de Amazon DocumentDB para la raíz. Cuenta de AWS
- AmazonDocDBReadOnlyAccess— Otorga acceso de solo lectura a todos los recursos de Amazon DocumentDB para la raíz. Cuenta de AWS
- AmazonDocDBConsoleFullAccess: otorga acceso completo para administrar Amazon DocumentDB y los recursos de clústeres elásticos de Amazon DocumentDB mediante la Consola de administración de AWS.
- AmazonDocDBElasticReadOnlyAccess— Concede acceso de solo lectura a todos los recursos del clúster elástico de Amazon DocumentDB para la raíz. Cuenta de AWS
- AmazonDocDBElasticFullAccess— Otorga acceso total a todos los recursos del clúster elástico de Amazon DocumentDB para la raíz. Cuenta de AWS

AmazonDocDBFullAccess

Esta política otorga permisos administrativos que brindan a una entidad principal acceso completo a todas las acciones de Amazon DocumentDB. Los permisos de esta política se agrupan de la siguiente manera:

- Los permisos de Amazon DocumentDB permiten todas las acciones de Amazon DocumentDB.
- Algunos de los permisos de Amazon EC2 de esta política son necesarios para validar los recursos aprobados en una solicitud de API. Esto es para garantizar que Amazon DocumentDB pueda utilizar correctamente los recursos con un clúster. El resto de los permisos de Amazon EC2 de esta política permiten a Amazon DocumentDB crear AWS los recursos necesarios para que pueda conectarse a los clústeres.

- Los permisos de Amazon DocumentDB se utilizan durante las llamadas a la API para validar los recursos transferidos en una solicitud. Son necesarios para que Amazon DocumentDB pueda utilizar la clave pasada con el clúster de Amazon DocumentDB.
- Los CloudWatch registros son necesarios para que Amazon DocumentDB pueda garantizar que se puede acceder a los destinos de entrega de los registros y que son válidos para su uso por parte de los agentes.

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

Esta política otorga permisos de solo lectura que permiten a los usuarios ver información en Amazon DocumentDB. Las entidades principales con esta política asociada no pueden realizar actualizaciones ni eliminar los recursos existentes, ni pueden crear nuevos recursos de Amazon

DocumentDB. Por ejemplo, las entidades principales con estos permisos pueden ver la lista de configuraciones y clústeres asociados a su cuenta, pero no pueden cambiar la configuración ni los ajustes de ningún clúster. Los permisos de esta política se agrupan de la siguiente manera:

- Los permisos de Amazon DocumentDB le permiten enumerar los recursos de Amazon DocumentDB, describirlos y obtener información sobre ellos.
- Los permisos de Amazon EC2 se utilizan para describir la VPC de Amazon, las subredes, los grupos de seguridad y las ENI asociadas a un clúster.
- El permiso de Amazon DocumentDB se utiliza para describir la clave asociada al clúster.

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

Otorga acceso total para administrar los recursos de Amazon DocumentDB de la Consola de administración de AWS siguiente manera:

- Los permisos de Amazon DocumentDB permiten todas las acciones de Amazon DocumentDB y de clústeres de Amazon DocumentDB.
- Algunos de los permisos de Amazon EC2 de esta política son necesarios para validar los recursos aprobados en una solicitud de API. Esto es para garantizar que Amazon DocumentDB pueda utilizar correctamente los recursos para ofrecer y mantener un clúster. El resto de los permisos de Amazon EC2 de esta política permiten a Amazon DocumentDB crear AWS los recursos necesarios para que pueda conectarse a sus clústeres, como VPCendpoint.
- AWS KMS los permisos se utilizan durante las llamadas a la API AWS KMS para validar los recursos aprobados en una solicitud. Son necesarios para que Amazon DocumentDB pueda utilizar la clave pasada para cifrar y descifrar los datos en reposo con el clúster elástico de Amazon DocumentDB.
- Los CloudWatch registros son necesarios para que Amazon DocumentDB pueda garantizar que se puede acceder a los destinos de entrega de los registros y que son válidos para auditar y crear perfiles para el uso de los registros.
- Los permisos de Secrets Manager son necesarios para validar un secreto determinado y usarlo para configurar el usuario administrador de los clústeres elásticos de Amazon DocumentDB.
- Se requieren permisos de Amazon RDS para las acciones de administración de clústeres de Amazon DocumentDB. En determinadas características de administración, Amazon DocumentDB utiliza una tecnología operativa que comparte con Amazon RDS.
- Los permisos de SNS permiten a las entidades principales acceder a las suscripciones y temas de Amazon Simple Notification Service (Amazon SNS), y publicar mensajes de Amazon SNS.
- Se requieren permisos de IAM para crear los roles vinculados al servicio necesarios para la publicación de métricas y registros.

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

Esta política otorga permisos de solo lectura que permiten a los usuarios ver información del clúster elástico en Amazon DocumentDB. Las entidades principales con esta política asociada no pueden realizar actualizaciones ni eliminar los recursos existentes, ni pueden crear nuevos recursos de Amazon DocumentDB. Por ejemplo, las entidades principales con estos permisos pueden ver la lista de configuraciones y clústeres asociados a su cuenta, pero no pueden cambiar la configuración ni los ajustes de ningún clúster. Los permisos de esta política se agrupan de la siguiente manera:

- Los permisos de clúster elástico de Amazon DocumentDB le permiten enumerar los recursos del clúster elástico de Amazon DocumentDB, describirlos y obtener información sobre ellos.
- CloudWatch los permisos se utilizan para verificar las métricas del servicio.

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

Esta política otorga permisos administrativos que brindan a una entidad principal acceso completo a todas las acciones de Amazon DocumentDB para un clúster elástico de Amazon DocumentDB.

Esta política usa AWS etiquetas (https://docs.aws.amazon.com/tag-editor/latest/userguide/tagging.html) dentro de las condiciones para determinar el acceso a los recursos. Si utiliza una secreta, debe estar etiquetada con una clave de etiqueta `DocDBElasticFullAccess` y un valor de etiqueta. Si utiliza una clave administrada por el cliente, debe estar etiquetada con una clave de etiqueta `DocDBElasticFullAccess` y un valor de etiqueta.

Los permisos de esta política se agrupan de la siguiente manera:

- Los permisos de clúster elástico de Amazon DocumentDB permiten todas las acciones de Amazon DocumentDB.
- Algunos de los permisos de Amazon EC2 de esta política son necesarios para validar los recursos aprobados en una solicitud de API. Esto es para garantizar que Amazon DocumentDB pueda utilizar correctamente los recursos para ofrecer y mantener un clúster. El resto de los permisos de Amazon EC2 de esta política permiten a Amazon DocumentDB crear AWS los recursos necesarios para que pueda conectarse a los clústeres como si fuera un punto de enlace de una VPC.
- AWS KMS se requieren permisos para que Amazon DocumentDB pueda usar la clave transferida para cifrar y descifrar los datos en reposo dentro del clúster elástico de Amazon DocumentDB.

> **Note**
>
> La clave administrada por el cliente debe tener una etiqueta con una clave `DocDBElasticFullAccess` y un valor de etiqueta.

- SecretsManager se requieren permisos para validar el secreto dado y usarlo para configurar al usuario administrador de los clústeres elásticos de Amazon DocumentDB.

> **Note**
>
> La secreta utilizada debe tener una etiqueta con una clave `DocDBElasticFullAccess` y un valor de etiqueta.

- Se requieren permisos de IAM para crear los roles vinculados al servicio necesarios para la publicación de métricas y registros.

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

No puede vincularse `AmazonDocDBElasticServiceRolePolicy` a sus AWS Identity and Access Management entidades. Esta política está adjunta a un rol vinculado a servicios que permite a Amazon DocumentDB realizar acciones en su nombre. Para obtener más información, consulte Service-linked funciones en los clústeres elásticos.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/DocDB-Elastic"
                    ]
                }
            }
        }
    ]
}
```

Amazon DocumentDB se actualiza a AWS políticas administradas

Cambio	Descripción	Fecha
AmazonDocDBElasticFullAccess, AmazonDocDBConsoleFullAccess - Cambio	Políticas actualizadas para añadir las acciones de mantenimiento pendientes.	2/11/2025
AmazonDocDBElasticFullAccess, AmazonDocDBConsoleFullAccess - Cambio	Se actualizaron las políticas para agregar start/stop y copiar las acciones de instantáneas del clúster.	2/21/2024
AmazonDocDBElasticReadOnlyAccess, AmazonDocDBElasticFullAccess - Cambio	Las políticas se actualizaron para añadir acciones `cloudwatch:GetMetricData` .	6/21/2023
AmazonDocDBElasticReadOnlyAccess: política nueva	Nueva política administrada para los clústeres elásticos de Amazon DocumentDB	6/8/2023

Cambio	Descripción	Fecha
AmazonDocDBElasticFullAccess: política nueva	Nueva política administrada para los clústeres elásticos de Amazon DocumentDB	6/5/2023
AmazonDocDB-ElasticServiceRolePolicy: política nueva	Amazon DocumentDB crea una nueva función vinculada a un AWS ServiceRoleForDocDB-Elastic servicio para los clústeres elásticos de Amazon DocumentDB.	11/30/2022
AmazonDocDBConsoleFullAccess: cambio	Política actualizada para añadir permisos de clúster elásticos y globales de Amazon DocumentDB.	11/30/2022
AmazonDocDBConsoleFullAccess, AmazonDocDBFullAccess, AmazonDocDBReadOnlyAccess: política nueva	Lanzamiento del servicio.	1/19/2017

Permisos de la API de Amazon DocumentDB: referencia de acciones, recursos y condiciones

Use las siguientes secciones como referencia cuando configure Uso de políticas basadas en identidad (políticas de IAM) para Amazon DocumentDB y escriba políticas de permisos que pueda asociar a una identidad de IAM (políticas basadas en identidad).

A continuación se lista cada una de las operaciones de la API de Amazon DocumentDB. En la lista se incluyen las acciones correspondientes a las que puede conceder permisos para realizar la acción, el AWS recurso para el que puede conceder los permisos y las claves de condición que puede incluir para un control de acceso detallado. Las acciones se especifican en el campo `Action` de la política, el valor del recurso en el campo `Resource` de la política y las condiciones en el

campo `Condition` de la política. Para obtener más información acerca de las condiciones, consulte Especificación de las condiciones de una política.

Puede usar claves AWS de condición generales en sus políticas de Amazon DocumentDB para expresar las condiciones. Para obtener una lista completa de las claves AWS generales, consulte las claves disponibles en la guía del usuario de IAM.

Puede probar las políticas de IAM con el simulador de política de IAM. Proporciona automáticamente una lista de los recursos y parámetros necesarios para cada AWS acción, incluidas las acciones de Amazon DocumentDB. El simulador de política de IAM determina los permisos necesarios para cada una de las acciones especificadas. Para obtener información sobre el simulador de políticas de IAM, consulte Probar las políticas de IAM con el simulador de políticas de IAM en la Guía del usuario de IAM.

> **Note**
>
> Para especificar una acción, use el prefijo `rds:` seguido del nombre de operación de la API (por ejemplo, `rds:CreateDBInstance`).

A continuación se muestran operaciones de API de Amazon RDS y sus acciones relacionadas, recursos y claves de condición.

Temas

- Acciones de Amazon DocumentDB que admiten permisos de nivel de recursos
- Acciones de Amazon DocumentDB que no admiten permisos de nivel de recursos

Acciones de Amazon DocumentDB que admiten permisos de nivel de recursos

Resource-level los permisos permiten especificar los recursos en los que los usuarios pueden realizar acciones. Amazon DocumentDB admite parcialmente los permisos de nivel de recursos. Esto significa que, en algunas acciones de Amazon DocumentDB, puede determinar cuándo se permite utilizarlas a los usuarios en función de si se cumplen una serie de condiciones o de los recursos concretos que pueden utilizar los usuarios. Por ejemplo, puede conceder a los usuarios permiso para modificar solo instancias específicas.

A continuación se muestran operaciones de API de Amazon DocumentDB y sus acciones relacionadas, recursos y claves de condición.

> **Note**
>
> En determinadas características de administración, Amazon DocumentDB utiliza una tecnología operativa que comparte con Amazon RDS. Para obtener más información sobre los permisos de Amazon DocumentDB, consulte Actions, resources, and condition keys for Amazon RDS en la Referencia de autorización de servicios.

Acciones y operaciones de la API de Amazon DocumentDB	Recursos	Claves de condición
AddTagsToResource `rds:AddTagsToResource`	Instancia `arn:aws:rds:` *`region`*`:`*`account-id`*` :db:`*`db-instance-name`*	`rds:db-tag`
	Grupo de subredes `arn:aws:rds:` *`region`*`:`*`account-id`*` :subgrp:`*`subnet-group-name`*	`rds:subgrp-tag`
ApplyPendingMaintenanceAction `rds:ApplyPendingMaintenanceAction`	Instancia `arn:aws:rds:` *`region`*`:`*`account-id`*` :db:`*`db-instance-name`*	`rds:db-tag`
CopyDBClusterSnapshot `rds:CopyDBClusterSnapshot`	Instantánea de clúster `arn:aws:rds:` *`region`*`:`*`account-id`*` :cluster-snapshot:`*`cluster-snapshot-name`*	`rds:cluster-snapshot-tag`
CreateDBCluster	Clúster	`rds:cluster-tag`

Acciones y operaciones de la API de Amazon DocumentDB	Recursos	Claves de condición
rds:CreateDBCluster	arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	
	Grupo de parámetros del clúster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
	Grupo de subredes arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
CreateDBClusterParameterGroup rds:CreateDBClusterParameterGroup	Grupo de parámetros del clúster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
CreateDBClusterSnapshot rds:CreateDBClusterSnapshot	Clúster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
	Instantánea de clúster arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag

Acciones y operaciones de la API de Amazon DocumentDB	Recursos	Claves de condición
CreateDBInstance rds:CreateDBInstance	Instancia arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
	Clúster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
CreateDBSubnetGroup rds:CreateDBSubnetGroup	Grupo de subredes arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DeleteDBInstance rds:DeleteDBInstance	Instancia arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
DeleteDBSubnetGroup rds:DeleteDBSubnetGroup	Grupo de subredes arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DescribeDBClusterParameterGroups rds:DescribeDBClusterParameterGroups	Grupo de parámetros del clúster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag

Acciones y operaciones de la API de Amazon DocumentDB	Recursos	Claves de condición
DescribeDBClusterParameters rds:DescribeDBClusterParameters	Grupo de parámetros del clúster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
DescribeDBclústers rds:DescribeDBClusters	Clúster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
DescribeDBClusterSnapshotAttributes rds:DescribeDBClusterSnapshotAttributes	Instantánea de clúster arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
DescribeDBSubnetGroups rds:DescribeDBSubnetGroups	Grupo de subredes arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DescribePendingMaintenanceActions rds:DescribePendingMaintenanceActions	Instancia arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag

Acciones y operaciones de la API de Amazon DocumentDB	Recursos	Claves de condición
FailoverDBCluster rds:FailoverDBCluster	Clúster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
ListTagsForResource rds:ListTagsForResource	Instancia arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
	Grupo de subredes arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
ModifyDBCluster rds:ModifyDBCluster	Clúster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
	Grupo de parámetros del clúster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
ModifyDBClusterParameterGroup rds:ModifyDBClusterParameterGroup	Grupo de parámetros del clúster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag

Acciones y operaciones de la API de Amazon DocumentDB	Recursos	Claves de condición
ModifyDBClusterSnapshotAttribute rds:ModifyDBClusterSnapshotAttribute	Instantánea de clúster arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
ModifyDBInstance rds:ModifyDBInstance	Instancia arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
RebootDBInstance rds:RebootDBInstance	Instancia arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
RemoveTagsFromResource rds:RemoveTagsFromResource	Instancia arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
	Grupo de subredes arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
ResetDBClusterParameterGroup rds:ResetDBClusterParameterGroup	Grupo de parámetros del clúster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag

Acciones y operaciones de la API de Amazon DocumentDB	Recursos	Claves de condición
RestoreDBClusterFromSnapshot rds:RestoreDBClusterFromSnapshot	Clúster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
	Instantánea de clúster arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
RestoreDBClusterToPointInTime rds:RestoreDBClusterToPointInTime	Clúster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
	Grupo de subredes arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag

Acciones de Amazon DocumentDB que no admiten permisos de nivel de recursos

Puede usar todas las acciones de Amazon DocumentDB de una política de IAM para conceder o denegar permiso a los usuarios para utilizar esa acción. Sin embargo, no todas las acciones de Amazon DocumentDB admiten permisos de nivel de recursos, que le permiten especificar los recursos en los que se puede realizar una acción. Las siguientes acciones de la API de Amazon DocumentDB no admiten actualmente permisos de nivel de recursos. Por lo tanto, para usar estas acciones en una política de IAM, debe conceder permiso a los usuarios para utilizar todos los recursos para la acción; para ello, emplee en su instrucción un carácter comodín * para el elemento `Resource`.

- `rds:DescribeDBClusterSnapshots`
- `rds:DescribeDBInstances`

Cross-service confusa prevención adjunta

El problema de la sustitución confusa es un problema de seguridad en el que una entidad que no tiene permiso para realizar una acción puede obligar a una entidad con más privilegios a realizar la acción. En AWS, la suplantación de identidad entre servicios puede resultar en el confuso problema de los diputados. Cross-service la suplantación de identidad puede ocurrir cuando un servicio (el servicio llamante) llama a otro servicio (el servicio llamado). El servicio que lleva a cabo las llamadas se puede manipular para utilizar sus permisos a fin de actuar en función de los recursos de otro cliente de una manera en la que no debe tener permiso para acceder. Para evitarlo, AWS proporciona herramientas que lo ayudan a proteger sus datos para todos los servicios con entidades principales de servicio a las que se les ha dado acceso a los recursos de su cuenta.

Utilice las claves de contexto de condición `aws:SourceAccount` global `aws:SourceArn` y las claves de contexto en las políticas de recursos para limitar los permisos que Amazon DocumentDB otorga a otro servicio al recurso. Utiliza `aws:SourceArn` si desea que solo se asocie un recurso al acceso entre servicios. Utiliza `aws:SourceAccount` si quiere permitir que cualquier recurso de esa cuenta se asocie al uso entre servicios.

La forma más eficaz de protegerse contra el problema de la sustitución confusa es utilizar la clave de contexto de condición global de `aws:SourceArn` con el ARN completo del recurso. Si no conoce el ARN completo del recurso o si está especificando varios recursos, utilice la clave de condición de contexto global `aws:SourceArn` con caracteres comodines (*) para las partes desconocidas del ARN. Por ejemplo, `arn:aws:`*`servicename`*`:*:`*`123456789012`*`:*`.

Si el valor de `aws:SourceArn` no contiene el ID de cuenta, como un ARN de bucket de Amazon S3, debe utilizar ambas claves de contexto de condición global para limitar los permisos.

El valor de `aws:SourceArn` debe ser clúster.

En el siguiente ejemplo se muestra cómo puede usar las claves de contexto de condición `aws:SourceAccount` global `aws:SourceArn` y las claves de contexto de Amazon DocumentDB para evitar el confuso problema de los sustitutos.

```
{
```

```
  "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"
      }
    }
  }
}
```

Autenticación mediante la identidad de IAM

Los usuarios y las aplicaciones de Amazon DocumentDB pueden utilizar los usuarios y roles de IAM para autenticarse en un clúster de Amazon DocumentDB. La autenticación de IAM de Amazon DocumentDB es un método de autenticación sin contraseña. Además, las aplicaciones cliente no envían los secretos de las contraseñas al clúster de Amazon DocumentDB cuando utilizan IAM. roles/users En su lugar, las conexiones de los clientes se autentican AWS STS mediante tokens de seguridad temporales. Non-administrative los usuarios y las aplicaciones ahora pueden usar el mismo ARN de identidad de IAM al conectarse a diferentes clústeres de Amazon DocumentDB y a otros servicios. AWS

También puede optar por utilizar la autenticación basada en contraseña y la autenticación de IAM para autenticar los usuarios y las aplicaciones en un clúster de Amazon DocumentDB. La autenticación de IAM solo está disponible en la versión 5.0 del clúster basado en instancias de Amazon DocumentDB. El usuario principal de Amazon DocumentDB no admite la autenticación de IAM mediante ARN de identidad de IAM.

> **Note**
>
> El usuario principal solo se puede autenticar mediante la autenticación basada en contraseñas existente.

Temas

- Introducción a la autenticación mediante roles y usuarios de IAM
- Configuración AWS tipos de cómputos para autenticarse en Amazon DocumentDB mediante AWS IAM
- Supervisión de solicitudes de autenticación de IAM
- Uso de la autenticación de IAM
- Controladores compatibles con IAM
- Preguntas frecuentes sobre la autenticación de identidad de IAM

Introducción a la autenticación mediante roles y usuarios de IAM

Los usuarios y roles de Amazon DocumentDB con identidades de IAM se crean y administran en una base de datos de `$external`.

Creación de un usuario

Conéctese como usuario principal y, a continuación, cree un usuario y un rol de IAM:

```
use $external;
db.createUser(
    {
        user: "arn:aws:iam::123456789123:user/iamuser",
        mechanisms: ["MONGODB-AWS"],
        roles: [ { role: "readWrite", db: "readWriteDB" } ]
    }
);
```

Como alternativa, agregue un usuario de Amazon DocumentDB con un rol de IAM:

```
use $external;
```

```
db.createUser(
    {
        user: "arn:aws:iam::123456789123:role/iamrole",
        mechanisms: ["MONGODB-AWS"],
        roles: [ { role: "readWrite", db: "readWriteDB" } ]
    }
);
```

Modificación de un usuario o rol de IAM

Modificación de un usuario de IAM existente:

```
use $external;
db.updateUser(
  "arn:aws:iam::123456789123:user/iamuser",
  {
    roles: [ { role: "read", db: "readDB" } ]
  }
);
```

Modificación de un rol de IAM existente:

```
use $external;
db.updateUser(
  "arn:aws:iam::123456789123:role/iamrole",
  {
    roles: [ { role: "read", db: "readDB" } ]
  }
);
```

Concesión o revocación de roles de un usuario de 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" } ]
);
```

Concesión o revocación de roles de un rol de 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" } ]
);
```

Eliminación de un usuario o un rol de IAM

Eliminación de un usuario de IAM existente:

```
use $external;
db.dropUser("arn:aws:iam::123456789123:user/iamuser");
```

Eliminación de un rol de IAM existente:

```
use $external;
db.dropUser("arn:aws:iam::123456789123:role/iamrole");
```

Configure un URI de conexión para autenticarse mediante IAM AWS

Para autenticarse mediante AWS IAM, utilice los siguientes parámetros de URI: `authSource` as y as. `$external` `authMechanism` `MONGODB-AWS` Si utiliza un usuario de IAM, los campos de nombre de usuario y contraseña se sustituyen por una clave de acceso y una clave secreta, respectivamente. Si asume un rol de IAM asociado al entorno en el que se encuentra (por ejemplo, función AWS Lambda , instancia de Amazon EC2). No es necesario pasar ninguna credencial de manera específica al autenticarse mediante el mecanismo `MONGODB-AWS`. Si utiliza controladores de MongoDB que admiten el mecanismo de autenticación `MONGODB-AWS`, los controladores también tienen la capacidad de recuperar las credenciales del rol de IAM de la instancia de procesamiento

(por ejemplo, Amazon EC2, la función de Lambda y otros). En el siguiente ejemplo, se utiliza un intérprete de comandos de mongo para autenticarse con `MONGODB-AWS` mediante el paso manual de una clave de acceso y una clave secreta (de un usuario de IAM) para demostrar la autenticación en Amazon DocumentDB.

En el siguiente ejemplo, se utiliza código Python para autenticarse con `MONGODB-AWS` sin pasar ninguna credencial de forma explícita (mediante un rol de IAM asociado al entorno) para demostrar la autenticación en 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')
```

En el siguiente ejemplo, se utiliza un intérprete de comandos de mongo para autenticarse con el mecanismo `MONGODB-AWS` mediante el paso manual de una clave de acceso y una clave secreta (de un usuario de IAM) para demostrar la autenticación en Amazon DocumentDB.

```
$ mongo 'mongodb://<access_key>:<secret_key>@<cluster_endpoint>:<db_port>/test?
authSource=%24external&authMechanism=MONGODB-AWS'
```

En el siguiente ejemplo, se utiliza un intérprete de comandos de mongo para autenticarse con `MONGODB-AWS` sin pasar ninguna credencial de forma explícita (mediante un rol de IAM asociado al entorno) para demostrar la autenticación en Amazon DocumentDB.

```
$ mongo 'mongodb://<cluster_endpoint>:<db_port>/test?authSource=
%24external&authMechanism=MONGODB-AWS'
```

Configuración AWS tipos de cómputos para autenticarse en Amazon DocumentDB mediante AWS IAM

Uso de Amazon EC2/AWS Lambda/AWS Fargate

Amazon EC2 usa las siguientes variables de entorno. Si tiene un rol de IAM asociado a la instancia de EC2 o un rol de IAM de ejecución asociado a una función de Lambda o una tarea de Amazon ECS, estas variables se rellenan automáticamente y el controlador puede obtener estos valores del entorno:

```
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
AWS_SESSION_TOKEN
```

Para obtener información sobre cómo usar las variables de entorno, consulte Utilice variables de entorno de Lambda en la Guía para desarrolladores de AWS Lambda .

Uso de Amazon EKS

Al asignar un rol a los pods de Amazon Elastic Kubernetes Service (Amazon EKS), se configurarán automáticamente las dos variables de entorno siguientes:

```
AWS_WEB_IDENTITY_TOKEN_FILE - path of web identity token file
AWS_ROLE_ARN - Name of IAM role to connect with
```

Con la ayuda de estas variables, asuma manualmente el rol a partir de su código mediante la llamada del AWS SDK para: `AssumeRoleWithWebIdentity`

- Omita el parámetro `ProviderID`.
- Busque el valor del parámetro `WebIdentityToken` en el archivo descrito en la variable de entorno `AWS_WEB_IDENTITY_TOKEN_FILE`.

Para obtener más información en Amazon EKS, consulte ¿Qué es Amazon EKS? en la Guía del usuario de Amazon EKS.

Supervisión de solicitudes de autenticación de IAM

Uso de la auditoría de Amazon DocumentDB

Ve a la carpeta de registro de auditoría de Amazon CloudWatch y usa diferentes patrones de búsqueda para obtener los registros para la autenticación de IAM. Por ejemplo, utilice `{ $.param.mechanism = "MONGODB-AWS" }` como patrón de búsqueda para “Buscar en todos los flujos de registro”.

Para obtener más información sobre los eventos compatibles en la auditoría, consulte Auditoría de eventos de Amazon DocumentDB.

Uso de las métricas de Amazon CloudWatch

`StsGetCallerIdentityCalls`: Esta métrica muestra cuántas `GetCallerIdentity` llamadas realiza una instancia de Amazon DocumentDB al punto de enlace regionalizado AWS Security Token Service (AWS STS). Consulte la especificación de `MONGODB-AWS` autenticación para saber por qué las instancias de base de datos deben realizar llamadas STS. `GetCallerIdentity`

Uso de la autenticación de IAM

Si no desea administrar el nombre de usuario y la contraseña de su propia base de datos, puede utilizar la autenticación de IAM. La autenticación de IAM solo está disponible en la versión 5.0 del clúster basado en instancias de Amazon DocumentDB.

La autenticación de IAM depende de AWS Security Token Service (AWS STS). Si recibes una excepción de AWS STS limitación al usar la autenticación de IAM, reduce la velocidad de conexión.

Para ver las cuotas de IAM, consulta IAM y las AWS STS cuotas en la Guía del usuario de IAM.

Controladores compatibles con IAM

Los controladores compatibles con Amazon DocumentDB 5.0 y el mecanismo de autenticación de `MONGODB-AWS` deberían funcionar con la implementación de autenticación de IAM en Amazon DocumentDB.

> **Important**
>
> Existe una limitación conocida en relación con Node.js los controladores anteriores a la versión 6.13.1, que actualmente no son compatibles con la autenticación de identidad de IAM en Amazon DocumentDB. Node.js Los controladores y las herramientas que utilizan el Node.js controlador (por ejemplo, mongosh) deben actualizarse para que utilicen Node.js la versión 6.13.1 o superior del controlador.

Preguntas frecuentes sobre la autenticación de identidad de IAM

¿Hay alguna muestra que pueda consultar?

Consulte estas páginas para ver ejemplos de casos de uso y configuraciones:

- Cómo los usuarios humanos pueden autenticarse en Amazon DocumentDB mediante usuarios y roles de IAM

- Password-less autenticación en Amazon DocumentDB mediante roles de IAM

Recibo un error al usar mi controlador de Python: «pymongo.errors». ConfigurationError: La autenticación de MONGODB requiere pymongo-auth-aws».AWS ¿Cómo puedo resolver esto?

Asegúrese de usar la siguiente declaración al instalar el controlador de Python con la autenticación de IAM:

```
pip install 'pymongo[aws]'
```

Esto instalará las AWS dependencias adicionales necesarias para que funcione la autenticación de IAM.

¿Se interrumpirá mi conexión cuando caduquen las credenciales temporales de mi rol de IAM?

No, las credenciales de IAM temporales solo se utilizan para establecer la conexión y la autenticación. A continuación, todas las autorizaciones adicionales se producen en el clúster de Amazon DocumentDB. Incluso con las credenciales de IAM rotate/expire, la conexión no se interrumpirá ni se estropeará.

Autenticación mediante SCRAM-SHA-256

Amazon DocumentDB (compatible con MongoDB) admite el mecanismo de autenticación Salted Challenge Response (SCRAM) para la autenticación basada en contraseñas. Con SCRAM, el servidor verifica que un cliente conozca la contraseña de un usuario sin transmitir la contraseña al servidor. Amazon DocumentDB admite dos mecanismos de SCRAM: y. `SCRAM-SHA-1` `SCRAM-SHA-256` `SCRAM-SHA-256`utiliza la función SHA-256 hash más potente y funciona con el modo FIPS.

Al crear o actualizar ese usuario, se asignan uno o ambos mecanismos de SCRAM a un usuario. Los usuarios que se crearon con `SCRAM-SHA-1` siguen funcionando sin ningún cambio. SCRAM-SHA-256 la autenticación está disponible en las versiones secundarias 5.0.1+ y 8.0.1+ de clústeres basados en instancias de Amazon DocumentDB.

> **La autenticación de SCRAM e IAM se excluyen mutuamente**
>
> Un usuario puede tener un mecanismo SCRAM o el `MONGODB-AWS` mecanismo, pero no ambos. Para obtener información sobre la autenticación de identidad de IAM, consulte. Autenticación mediante la identidad de IAM

Temas

- Creación de usuarios con mecanismos de SCRAM
- Actualización de los usuarios existentes
- Autenticando con SCRAM-SHA-256

Creación de usuarios con mecanismos de SCRAM

Al crear un usuario, puede especificar los mecanismos de SCRAM que desea habilitar en el `mechanisms` campo. Los ejemplos siguientes crean usuarios con `SCRAM-SHA-1` solo, `SCRAM-SHA-256` solo y ambos mecanismos:

```
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']
});
```

Mecanismos predeterminados

Si no especifica el `mechanisms` campo, Amazon DocumentDB habilita ambos campos `SCRAM-SHA-1` y `SCRAM-SHA-256` para el usuario. Los dos comandos siguientes son equivalentes:

```
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']
});
```

Al habilitar ambos mecanismos de forma predeterminada, se mantiene la compatibilidad con los controladores MongoDB más antiguos, que solo los admiten. `SCRAM-SHA-1` Se prefieren los controladores más nuevos y el modo FIPS. `SCRAM-SHA-256` Si no quieres que ningún usuario se autentique`SCRAM-SHA-1`, especifícalo solo `SCRAM-SHA-256` en el `mechanisms` campo para todos tus usuarios.

Actualización de los usuarios existentes

Los usuarios creados con `SCRAM-SHA-1` siguen siendo compatibles sin cambios. Para cambiar un usuario existente de `SCRAM-SHA-1` a`SCRAM-SHA-256`, actualice el usuario con una contraseña nueva y el `SCRAM-SHA-256` mecanismo:

```
db.runCommand({
    updateUser: 'existing',
    pwd: passwordPrompt(),
    mechanisms: ['SCRAM-SHA-256']
});
```

Autenticando con SCRAM-SHA-256

El controlador se autentica automáticamente mediante los parámetros de la cadena de conexión antes de enviar el primer comando en cada conexión. Para seleccionar `SCRAM-SHA-256` un URI de conexión, defina `SCRAM-SHA-256` el `authMechanism` parámetro en:

```
mongodb://username:password@hostname:port/authenticationDatabase?authMechanism=SCRAM-
SHA-256
```

También puedes enviar una solicitud de autenticación explícita desde mongosh:

```
db.auth({user: 'sha256user', pwd: passwordPrompt(), mechanism: 'SCRAM-SHA-256'})
```

```
Enter password:
1
```

Si no especificas un mecanismo de autenticación en tus opciones de conexión, el controlador selecciona automáticamente un mecanismo compatible con el usuario de la base de datos de destino.

Administración de usuarios de Amazon DocumentDB

En Amazon DocumentDB, los usuarios se autentican en un clúster junto con una contraseña. Cada clúster tiene credenciales de inicio de sesión principales que se establecen durante la creación del clúster.

> **Note**
>
> Todos los nuevos usuarios creados antes del 26 de marzo de 2020 han recibido los roles `dbAdminAnyDatabase`, `readWriteAnyDatabase`, y `clusterAdmin`. Se recomienda que vuelva a evaluar todos los usuarios y modifique los roles según sea necesario para aplicar los privilegios mínimos para todos los usuarios de los clústeres.
> Para obtener más información, consulte Acceso a bases de datos mediante Role-Based Access Control.

Usuario principal y **`administrador de servicio`**

Un clúster recién creado de Amazon DocumentDB tiene dos usuarios: el usuario principal y el usuario `serviceadmin`.

El usuario principal es un usuario único con privilegios que puede realizar tareas administrativas y crear usuarios adicionales con roles. Cuando se conecta a un clúster de Amazon DocumentDB por primera vez, debe autenticarse con las credenciales de inicio de sesión principales. El usuario principal recibe estos permisos administrativos para un clúster de Amazon DocumentDB cuando se crea ese clúster y se le concede el rol de `root`.

El usuario `serviceadmin` se crea implícitamente cuando se crea el clúster. Todos los clústeres de Amazon DocumentDB tienen un `serviceadmin` usuario que permite administrar AWS el clúster. No puede iniciar sesión como, ni eliminarlo, cambiarlo de nombre, cambiar su contraseña o cambiar los

permisos para `serviceadmin`. Cualquier intento de realizar una de estas operaciones producirá un error.

> **Note**
>
> El principal y los usuarios `serviceadmin` de un clúster de Amazon DocumentDB no se pueden eliminar y el rol de `root` para el usuario principal no se puede revocar.
> Si olvida su contraseña de usuario principal, puede restablecerla utilizando el Consola de administración de AWS o el AWS CLI.

Creación de usuarios adicionales

Después de conectarse como usuario principal (o cualquier usuario que tenga el rol `createUser`), puede crear un nuevo usuario, como se muestra a continuación.

```
db.createUser(
    {
        user: "sample-user-1",
        pwd: "password123",
        roles:
            [{"db":"admin", "role":"dbAdminAnyDatabase" }]
    }
)
```

Para ver los detalles del usuario, puede utilizar el comando `show users` de la siguiente manera. También puede eliminar usuarios con el comando `dropUser`. Para obtener más información, consulte Comandos comunes.

```
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"
     }
    ]
}
```

En este ejemplo, el nuevo usuario `sample-user-1` se atribuye a la base de datos `admin`. Esto se aplica siempre en el caso de un usuario nuevo. Amazon DocumentDB no tiene el concepto de una `authenticationDatabase` y, por lo tanto, toda la autenticación se realiza en el contexto de la base de datos `admin`.

Al crear usuarios, si omite el campo db al especificar el rol, Amazon DocumentDB atribuirá implícitamente el rol a la base de datos en la que se está emitiendo la conexión. Por ejemplo, si la conexión se emite contra la base de datos `sample-database` y ejecuta el siguiente comando, el usuario `sample-user-2` se creará en la base de datos del `admin` y tendrá permisos `readWrite` para la base de datos `sample-database`.

```
db.createUser(
    {
        user: "sample-user-2",
```

```
        pwd: "password123",
        roles:
            ["readWrite"]
    }
)
```

La creación de usuarios con roles cuyo ámbito se encuentra en todas las bases de datos (por ejemplo, `readInAnyDatabase`) requiere que se encuentre en el contexto de la base de datos `admin` al crear el usuario o que indique explícitamente la base de datos para el rol al crear el usuario.

Para cambiar el contexto de la base de datos, puede utilizar el siguiente comando.

```
use admin
```

Para obtener más información sobre el control de acceso basado en roles y la aplicación de privilegios mínimos entre los usuarios del clúster, consulte Acceso a bases de datos mediante Role-Based Access Control.

Rotación automática de contraseñas para Amazon DocumentDB

Con AWS Secrets Manager, puedes reemplazar las credenciales codificadas de tu código (incluidas las contraseñas) por una llamada a la API a Secrets Manager para recuperar el secreto mediante programación. Esto ayuda a garantizar la integridad del secreto si alguien examina el código, dado que el secreto sencillamente no está allí. Asimismo, puede configurar Secrets Manager para rotar el secreto automáticamente de acuerdo con la programación que especifique. Esto le permite reemplazar secretos a largo plazo con secretos a corto plazo, lo que contribuye a reducir significativamente el riesgo de peligro.

Con Secrets Manager, puede rotar automáticamente sus contraseñas de Amazon DocumentDB (es decir, las contraseñas secretas) mediante una AWS Lambda función que proporciona Secrets Manager.

Para obtener más información sobre la integración nativa con Amazon DocumentDB AWS Secrets Manager y su integración nativa, consulte lo siguiente:

- Blog: How to rotate Amazon DocumentDB and Amazon Redshift credentials in AWS Secrets Manager
- ¿Qué es AWS Secrets Manager?

- Rotación de secretos de AWS Secrets Manager
- Credenciales de Amazon DocumentDB en Secrets Manager

Acceso a bases de datos mediante Role-Based Access Control

Puede restringir el acceso a las acciones que los usuarios pueden realizar en bases de datos mediante el control de acceso basado en roles (RBAC) en Amazon DocumentDB (compatible con MongoDB). RBAC funciona otorgando uno o más roles a un usuario. Estos roles determinan las operaciones que un usuario puede realizar en los recursos de la base de datos. Actualmente, Amazon DocumentDB admite tanto roles integrados que se centran en el nivel de base de datos, como `read`, `readWrite`, `readAnyDatabase`, `clusterAdmin`, y roles definidos por el usuario que se pueden limitar a acciones específicas y recursos granulares, como colecciones, en función de sus requisitos.

Los casos de uso comunes para RBAC incluyen la imposición de privilegios mínimos mediante la creación de usuarios con acceso de sólo lectura a las bases de datos o colecciones en un clúster, y diseños de aplicaciones multitenant que permiten a un solo usuario acceder a una base de datos determinada o colección en un clúster.

> **Note**
>
> Todos los nuevos usuarios creados antes del 26 de marzo de 2020 han recibido los roles `dbAdminAnyDatabase`, `readWriteAnyDatabase`, y `clusterAdmin`. Se recomienda volver a evaluar todos los usuarios existentes y modificar los roles según sea necesario para aplicar los privilegios mínimos para los clústeres.

Temas

- Conceptos de RBAC
- Introducción a los roles integrados de RBAC
- Introducción a los roles definidos por el usuario de RBAC
- Conexión a Amazon DocumentDB como un usuario
- Comandos comunes
- Diferencias funcionales
- Límites

- Acceso a la base de datos mediante el control de Role-Based acceso

Conceptos de RBAC

Los siguientes son términos y conceptos importantes relacionados con el control de acceso basado en roles. Para obtener más información sobre los usuarios de Amazon DocumentDB, consulteAdministración de usuarios de Amazon DocumentDB.

- Usuario: entidad individual que puede autenticarse en la base de datos y realizar operaciones.
- Contraseña: secreto que se utiliza para autenticar al usuario.
- Rol: autoriza a un usuario a realizar acciones en una o más bases de datos.
- Base de datos de administración: la base de datos en la que se almacenan los usuarios y en la que se autoriza su uso.
- Base de datos (**db**): el espacio de nombres dentro de los clústeres que contiene colecciones para almacenar documentos.

El comando siguiente crea un usuario llamado `sample-user`.

```
db.createUser({user: "sample-user", pwd: "abc123", roles: [{role: "read", db: "sample-database"}]})
```

En este ejemplo:

- `user: "sample-user"`: indica el nombre de usuario.
- `pwd: "abc123"`: indica la contraseña del usuario.
- `role: "read", "db: "sample-database"`: indica que el `sample-user` de usuario tendrá permisos de lectura en `sample-database`.



El siguiente ejemplo muestra el resultado después de obtener el usuario `sample-user` con `db.getUser(sample-user)`. En este ejemplo, el usuario `sample-user` reside en la base de datos del `admin` pero tiene el rol de lectura para la base de datos `sample-database`.



Al crear usuarios, si omite el campo db al especificar el rol, Amazon DocumentDB atribuirá implícitamente el rol a la base de datos en la que se está emitiendo la conexión. Por ejemplo, si la conexión se emite contra la base de datos `sample-database` y ejecuta el siguiente comando, el usuario `sample-user` se creará en la base de datos del `admin` y tendrá permisos `readWrite` para la base de datos `sample-database`.

```
db.createUser({user: "sample-user", pwd: "abc123", roles: ["readWrite"]})
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
   "user":"sample-user",
   "roles":[
      {
         "db":"sample-database",
         "role":"readWrite"
      }
   ]
}
```

La creación de usuarios con roles cuyo ámbito se encuentra en todas las bases de datos (por ejemplo, `readAnyDatabase`) requiere que esté en el contexto de la base de datos de `admin` al crear el usuario o que indique explícitamente la base de datos para el rol al crear el usuario. Para emitir comandos contra la base de datos del `admin`, puede utilizar el comando `use admin`. Para obtener más información, consulte Comandos comunes.

Introducción a los roles integrados de RBAC

Para ayudarle a comenzar con el control de acceso basado en roles, esta sección le guía a través de un escenario de ejemplo de aplicación de privilegios mínimos mediante la creación de roles para tres usuarios con funciones de trabajo diferentes.

- `user1` es un nuevo administrador que necesita poder ver y acceder a todas las bases de datos de un clúster.
- `user2` es un nuevo empleado que necesita acceso a una sola base de datos, `sample-database-1`, en ese mismo clúster.
- `user3` es un empleado existente que necesita ver y tener acceso a una base de datos diferente, `sample-database-2`, a la que no tenían acceso antes, en el mismo clúster.

En un momento posterior, tanto `user1` como `user2` abandonan la empresa y por lo tanto su acceso debe ser revocado.

Para crear usuarios y otorgar roles, el usuario con el que se autentique en el clúster debe tener un rol asociado que pueda realizar acciones para `createUser` y `grantRole`. Por ejemplo, los roles `admin` y `userAdminAnyDatabase` pueden otorgar tales habilidades, por ejemplo. Para ver las acciones por rol, consulte Acceso a la base de datos mediante el control de Role-Based acceso.

> **Note**
>
> En Amazon DocumentDB, todas las operaciones de usuario y rol (por ejemplo, `create`, `get`, `drop`, `grant`, `revoke`, etc.) se realizan implícitamente en la base de datos de `admin`, independientemente de que se estén ejecutando o no comandos contra la base de datos `admin`.

En primer lugar, para comprender cuáles son los usuarios y roles actuales en el clúster, puede ejecutar el comando `show users`, como en el ejemplo siguiente. Verá dos usuarios `serviceadmin` y el usuario principal para el clúster. Estos dos usuarios siempre existen y no se pueden eliminar. Para obtener más información, consulte Administración de usuarios de Amazon DocumentDB.

```
show users
```

Para `user1`, cree un rol con acceso de lectura y escritura a todas las bases de datos de todo el clúster con el siguiente comando.

```
db.createUser({user: "user1", pwd: "abc123", roles: [{role: "readWriteAnyDatabase", db:
 "admin"}]})
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
   "user":"user1",
   "roles":[
      {
         "role":"readWriteAnyDatabase",
         "db":"admin"
      }
   ]
}
```

Para `user2`, cree un rol con acceso de sólo lectura a la base de datos `sample-database-1` con el siguiente comando.

```
db.createUser({user: "user2", pwd: "abc123", roles: [{role: "read", db: "sample-
database-1"}]})
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
   "user":"user2",
   "roles":[
      {
         "role":"read",
         "db":"sample-database-1"
      }
   ]
}
```

Para simular el escenario en que `user3` es un usuario existente, primero cree el usuario `user3` y a continuación asigne un nuevo rol a `user3`.

```
db.createUser({user: "user3", pwd: "abc123", roles: [{role: "readWrite", db: "sample-
database-1"}]})
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
   "user":"user3",
   "roles":[
      {
         "role":"readWrite",
         "db":"sample-database-1"
      }
   ]
}
```

Ahora que se ha creado el usuario `user3`, asigne a `user3` el rol `read` a `sample-database-2`.

```
db.grantRolesToUser("user3", [{role: "read", db: "sample-database-2"}])
```

Por último, tanto `user1` como `user2` abandonan la empresa y su acceso al clúster debe revocarse. Puede hacer esto dejando caer a los usuarios, de la siguiente manera.

```
db.dropUser("user1")
db.dropUser("user2")
```

Para asegurarse de que todos los usuarios tienen los roles adecuados, puede enumerar todos los usuarios con el siguiente comando.

```
show users
```

La salida de esta operación será similar a lo que se indica a continuación.

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

Introducción a los roles definidos por el usuario de RBAC

Para ayudarle a comenzar con roles definidos por el usuario, esta sección le guía a través de un escenario de ejemplo de aplicación de privilegios mínimos mediante la creación de roles para tres usuarios con funciones distintas.

En este ejemplo, se aplica lo siguiente:

- `user1` es un nuevo administrador que necesita poder ver y acceder a todas las bases de datos de un clúster.
- `user2` es un nuevo empleado que necesita acceso a la acción ´encontrar´ en una sola base de datos, `sample-database-1`, en ese mismo clúster.

- `user3` es un empleado existente que necesita ver y tener acceso a una colección específica, col2 en una base de datos distinta, `sample-database-2` a la que no tenía acceso antes, en el mismo clúster.
- Para `user1`, cree un rol con acceso de lectura y escritura a todas las bases de datos de todo el clúster con el siguiente comando.

```
db.createUser(
 {
     user: "user1", pwd: "abc123",
     roles: [{role: "readWriteAnyDatabase", db: "admin"}]
 }
 )
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
   "user":"user1",
   "roles":[
      {
         "role":"readWriteAnyDatabase",
         "db":"admin"
      }
   ]
}
```

Para `user2`, cree un rol con privilegios de “búsqueda” para todas las colecciones de la base de datos `sample-database-1` con el siguiente comando. Tenga en cuenta que este rol garantizaría que los usuarios asociados solo puedan ejecutar consultas de búsqueda.

```
db.createRole(
{
    role: "findRole",
    privileges: [
    {
        resource: {db: "sample-database-1", collection: ""}, actions: ["find"]
    }],
    roles: []
}
)
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
   "role":"findRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-1",
            "collection":""
         },
         "actions":[
            "find"
         ]
      }
   ],
   "roles":[

   ]
}
```

A continuación, cree el usuario (`user2`) y adjunte el rol `findRole` creado recientemente al usuario.

```
db.createUser(
{
    user: "user2",
    pwd: "abc123",
    roles: []
})

 db.grantRolesToUser("user2",["findRole"])
```

Para simular el escenario en que `user3` es un usuario existente, primero cree el usuario `user3` y a continuación asigne un nuevo rol denominado collectionRole a `user3`, algo que haremos en el próximo paso.

Ahora puede asignar un nuevo rol al `user3`. Este nuevo rol permitirá al `user3` insertar, actualizar, eliminar y acceder a una colección específica col2 en la `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: []
}
)
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
   "role":"collectionRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-2",
            "collection":"col2"
         },
         "actions":[
            "find",
            "update",
            "insert",
            "remove"
         ]
      }
   ],
   "roles":[

   ]
}
```

Ahora que se ha creado el usuario `user3`, asigne a `user3` el rol `collectionFind`.

```
db.grantRolesToUser("user3",["collectionRole"])
```

Por último, tanto `user1` como `user2` abandonan la empresa y su acceso al clúster debe revocarse. Puede hacer esto dejando caer a los usuarios, de la siguiente manera.

```
db.dropUser("user1")
db.dropUser("user2")
```

Para asegurarse de que todos los usuarios tienen los roles adecuados, puede enumerar todos los usuarios con el siguiente comando.

```
show users
```

La salida de esta operación será similar a lo que se indica a continuación.

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

Conexión a Amazon DocumentDB como un usuario

Cuando se conecta a un clúster de Amazon DocumentDB, se conecta en el contexto de una base de datos determinada. De forma predeterminada, si no especifica una base de datos en la cadena de conexión, se conectará automáticamente al clúster en el contexto de la base de datos de `test`. Todos los comandos de nivel de recopilación como `insert` y `find` se emiten contra colecciones en la base de datos `test`.

Para ver la base de datos en la que se encuentra o, en otras palabras, en la que está emitiendo comandos, utilice el comando db del intérprete de comandos mongo, de la siguiente manera.

Consulta:

```
db
```

Salida:

```
test
```

Aunque la conexión predeterminada puede estar en el contexto de la base de datos de `test`, eso no significa necesariamente que el usuario asociado a la conexión esté autorizado a realizar acciones en la base de datos de `test`. En el escenario de ejemplo anterior, si se autentica como el usuario `user3`, que tiene el rol `readWrite` de la base de datos `sample-database-1`, el contexto predeterminado de la conexión es la base de datos de `test`. Sin embargo, si intenta insertar un documento en una colección de la base de datos de `test`, recibirá un mensaje de error Authorization failure (Error de autorización). Esto se debe a que ese usuario no está autorizado a realizar ese comando en esa base de datos, como se muestra a continuación.

Consulta:

```
db
```

Salida:

```
test
```

Consulta:

```
db.col.insert({x:1})
```

Salida:

```
WriteCommandError({ "ok" : 0, "code" : 13, "errmsg" : "Authorization failure" })
```

Si cambia el contexto de la conexión a la base de datos `sample-database-1`, puede escribir en la colección para la que el usuario tiene la autorización para hacerlo.

Consulta:

```
use sample-database-1
```

Salida:

```
switched to db sample-database-1
```

Consulta:

```
db.col.insert({x:1})
```

Salida:

```
WriteResult({ "nInserted" : 1})
```

Cuando se autentica en un clúster con un usuario determinado, también puede especificar la base de datos en la cadena de conexión. Al hacerlo, se elimina la necesidad de realizar el comando `use` después de que el usuario haya sido autenticado en la base de datos del `admin`.

La siguiente cadena de conexión autentica al usuario en la base de datos `admin`, pero el contexto de la conexión estará en la base de datos `sample-database-1`.

```
mongo "mongodb://user3:abc123@sample-cluster.node.us-east-1.docdb.amazonaws.com:27017/
sample-database-2"
```

Comandos comunes

Esta sección proporciona ejemplos de comandos comunes que utilizan el control de acceso basado en roles en Amazon DocumentDB. Debe estar en el contexto de la base de datos del `admin` para crear y modificar usuarios y roles. Puede utilizar el comando `use admin` para cambiar a la base de datos del `admin`.

> **Note**
>
> Las modificaciones a los usuarios y roles se producirán implícitamente en la base de datos del `admin`. La creación de usuarios con roles que tienen un ámbito en todas las bases de datos (por ejemplo, `readAnyDatabase`) requiere que esté en el contexto de la base de datos del `admin` (es decir, `use admin`) al crear el usuario, o que indique explícitamente la base de datos para el rol al crear el usuario (como se muestra en el Ejemplo 2 en este).

Ejemplo 1: crear un usuario con el rol `read` para la base de datos `foo`.

```
db.createUser({user: "readInFooBar", pwd: "abc123", roles: [{role: "read", db:
 "foo"}]})
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
   "user":"readInFooBar",
   "roles":[
      {
         "role":"read",
         "db":"foo"
      }
   ]
}
```

Ejemplo 2: crear un usuario con acceso de lectura en todas las bases de datos.

```
db.createUser({user: "readAllDBs", pwd: "abc123", roles: [{role: "readAnyDatabase", db:
 "admin"}]})
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
   "user":"readAllDBs",
   "roles":[
      {
         "role":"readAnyDatabase",
         "db":"admin"
      }
   ]
}
```

Ejemplo 3: otorgar el rol `read` a un usuario existente en una nueva base de datos.

```
db.grantRolesToUser("readInFooBar", [{role: "read", db: "bar"}])
```

Ejemplo 4: actualizar el rol de un usuario.

```
db.updateUser("readInFooBar", {roles: [{role: "read", db: "foo"}, {role: "read", db:
 "baz"}]})
```

Ejemplo 5: revocar el acceso a una base de datos de un usuario.

```
db.revokeRolesFromUser("readInFooBar", [{role: "read", db: "baz"}])
```

Ejemplo 6: describir un rol integrado.

```
db.getRole("read", {showPrivileges:true})
```

La salida de esta operación será similar a lo que se indica a continuación.

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

Ejemplo 7: eliminar un usuario del clúster.

```
db.dropUser("readInFooBar")
```

La salida de esta operación será similar a lo que se indica a continuación.

```
true
```

Ejemplo 8: crear un rol con acceso de lectura y escritura a una colección específica

```
db.createRole(
{
    role: "collectionRole",
    privileges: [
    {
        resource: {db: "sample-database-2", collection: "col2"}, actions: ["find",
 "update", "insert", "remove"]
    }],
    roles: []
}
)
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
   "role":"collectionRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-2",
            "collection":"col2"
         },
         "actions":[
            "find",
            "update",
            "insert",
            "remove"
         ]
      }
   ],
   "roles":[

   ]
}
```

Ejemplo 9: crear un usuario y asignar un rol definido por el usuario

```
db.createUser(
{
    user: "user3",
    pwd: "abc123",
    roles: []
```

```
})

db.grantRolesToUser("user3",["collectionRole"])
```

Ejemplo 10: otorgar privilegios adicionales a un rol definido por el usuario

```
db.grantPrivilegesToRole(
  "collectionRole",
  [
    {
      resource: { db: "sample-database-1", collection: "col1" },
      actions: ["find", "update", "insert", "remove"]
    }
  ]
)
```

Ejemplo 11: eliminar privilegios de un rol definido por el usuario

```
db.revokePrivilegesFromRole(
  "collectionRole",
  [
    {
      resource: { db: "sample-database-1", collection: "col2" },
      actions: ["find", "update", "insert", "remove"]
    }
  ]
)
```

Ejemplo 12: actualizar un rol definido por el usuario existente

```
db.updateRole(
  "collectionRole",
  {
    privileges: [
    {
        resource: {db: "sample-database-3", collection: "sample-collection-3"},
 actions: ["find", "update", "insert", "remove"]
    }],
    roles: []
  }
)
```

Diferencias funcionales

En Amazon DocumentDB, las definiciones de usuario y rol se almacenan en la base de datos del `admin` y los usuarios se autentican en la base de datos del `admin`. Esta funcionalidad difiere de la MongoDB Community Edition, pero es consistente con MongoDB Atlas.

Amazon DocumentDB también admite flujos de cambios, lo que brinda una secuencia en orden cronológico de los eventos de actualización que se producen dentro de las colecciones de su clúster. La acción `listChangeStreams` se aplica en el nivel de clúster (es decir, en todas las bases de datos) y la acción `modifyChangeStreams` puede aplicarse en el nivel de base de datos y en el nivel de clúster.

Límites

La siguiente tabla contiene los límites del control de Role-Based acceso en Amazon DocumentDB.

Description (Descripción)	Límite
Número de usuarios por clúster	1 000
Número de roles asociados a un usuario	1 000
Número de roles definidos por el usuario	100
Número de recursos asociados a un privilegio	100

Acceso a la base de datos mediante el control de Role-Based acceso

Con el control de acceso basado en roles, puede crear un usuario y otorgarle uno o más roles para determinar qué operaciones puede realizar el usuario en una base de datos o clúster.

A continuación se muestra una lista de roles integrados que se admiten actualmente en Amazon DocumentDB.

> **Note**
>
> En Amazon DocumentDB 4.0 y 5.0, los comandos `ListCollection` y `ListDatabase` pueden usar, de forma opcional, los parámetros `authorizedCollections` y `authorizedDatabases` para enumerar las colecciones y bases de datos a las que el

usuario tiene permiso para acceder, con los roles `listCollections` y `listDatabase`, respectivamente. Ahora, los usuarios pueden eliminar sus propios cursores sin necesitar el rol `KillCursor`.

Database user

Nombre del rol	Description (Descripción)	Acciones

`read`	Otorga a un usuario acceso de lectura a la base de datos especificada.	`changeStreams`
		`collStats`
		`dbStats`
		`find`
		`killCursors`
		`listIndexes`
		`listCollections`

`readWrite`	Otorga al usuario acceso de lectura y escritura a la base de datos especificada.	Todas las acciones de los permisos `read`.
		`createCollection`
		`dropCollection`
		`createIndex`
		`dropIndex`
		`insert`
		`killCursors`

Nombre del rol	Description (Descripción)	Acciones

		`listIndexes`
		
		`listColle ctions`
		
		`remove`
		
		`update`

Cluster user

Nombre del rol	Description (Descripción)	Acciones

leer AnyDatabase	Otorga a un usuario acceso de lectura a todas las bases de datos del clúster.	Todas las acciones de los permisos `read`.
		
		`listChang eStreams`
		
		`listDatabases`

`readWrite AnyDatabase`	Otorga a un usuario accesos de lectura y escritura a todas las bases de datos del clúster.	Todas las acciones de los permisos `readWrite` .
		
		`listChang eStreams`
		
		`listDatabases`

`userAdmin AnyDatabase`	Otorga al usuario la capacidad de asignar y modificar los roles o privilegios que cualquier usuario tiene en la base de datos especific ada.	`changeCus tomData`
		
		`changePassword`
		
		`createUser`

Nombre del rol	Description (Descripción)	Acciones

		`dropRole`
		`dropUser`
		`grantRole`
		`listDatabases`
		`revokeRole`
		`viewRole`
		`viewUser`

`dbAdminAnyDatabase`	Otorga al usuario la capacidad de realizar funciones de administración de bases de datos en cualquier base de datos especificada.	Todas las acciones de los permisos `dbAdmin`.
		`dropCollection`
		`listDatabases`
		`listChangeStreams`
		`modifyChangeStreams`

Superuser

Nombre del rol	Description (Descripción)	Acciones
`root`	Otorga a un usuario acceso a los recursos y operaciones de todos los siguientes roles combinados: `readWriteAnyDatabase` , `dbAdminAnyDatabase` , `userAdminAnyDatabase` , `clusterAdmin` , `restore`, y `backup`.	Todas las acciones de `readWriteAnyDatabase` , `dbAdminAnyDatabase`

Nombre del rol	Description (Descripción)	Acciones
		, `userAdmin` `AnyDatabase` , `clusterAdmin` , `restore`, y `backup`.

Database administrator

Nombre del rol	Description (Descripción)	Acciones

`dbAdmin`	Otorga al usuario la capacidad de realizar tareas administrativas en la base de datos especificada.	`bypassDocumentValidation`
		`collMod`
		`collStats`
		`createCollection`
		`createIndex`
		`dropCollection`
		`dropDatabase`
		`dropIndex`
		`dbStats`
		`find`
		`killCursors`
		`listIndexes`
		`listCollections`

Nombre del rol	Description (Descripción)	Acciones
		`modifyChangeStreams`
`dbOwner`	Otorga al usuario la capacidad de realizar cualquier tarea administrativa en la base de datos especificada combinando los roles `dbAdmin` y `readWrite` .	Todas las acciones de `dbAdmin` y `readWrite` .

Cluster administrator

Role name (Nombre de rol)	Description (Descripción)	Acciones

`clusterAdmin`	Otorga a un usuario el mayor acceso a la administración de clústeres mediante la combinación de los roles `clusterManager` , `clusterMonitor` y `hostManager` .	Todas las acciones de `clusterManager` , `clusterMonitor` y `hostManager` .
		
		`listChangeStreams`
		
		`dropDatabase`
		
		`modifyChangeStreams`

`clusterManager`	Otorga al usuario la capacidad de realizar acciones de administración y supervisión en el clúster especificado.	`listChangeStreams`
		
		`listSessions`
		
		`modifyChangeStreams`

Role name (Nombre de rol)	Description (Descripción)	Acciones
		`replSetGetConfig`

`clusterMonitor`	Otorga a un usuario la capacidad de tener acceso de solo lectura a las herramientas de supervisión.	`collStats`
		`dbStats`
		`find`
		`getParameter`
		`hostInfo`
		`indexStats`
		`killCursors`
		`listChangeStreams`
		`listCollections`
		`listDatabases`
		`listIndexes`
		`listSessions`
		`replSetGetConfig`
		`serverStatus`
		`top`

Role name (Nombre de rol)	Description (Descripción)	Acciones

`hostManager`	Otorga al usuario la capacidad de supervisar y administrar servidores.	`auditConfigure`
		`killCursors`
		`killAnyCursor`
		`killAnySession`
		`killop`

Backup administrator

Nombre del rol	Description (Descripción)	Acciones

`backup`	Otorga a un usuario el acceso necesario para realizar copias de seguridad de los datos.	`getParameter`
		`insert`
		`find`
		`listChangeStreams`
		`listCollections`
		`listDatabases`
		`listIndexes`
		`update`

`restore`	Otorga a un usuario el acceso necesario para restaurar los datos.	`bypassDocumentValidation`

Nombre del rol	Description (Descripción)	Acciones
		changeCustomData changePassword collMod createCollection createIndex createUser dropCollection dropRole dropUser getParameter grantRole find insert listCollections modifyChangeStreams revokeRole remove viewRole viewUser

Nombre del rol	Description (Descripción)	Acciones
		`update`

Actualización de los certificados TLS de Amazon DocumentDB

Temas

- Actualización de la aplicación y del clúster de Amazon DocumentDB
- Rotación automática de certificados del servidor
- Resolución de problemas
- Preguntas frecuentes

El certificado de la autoridad de certificación (CA) para los clústeres de Amazon DocumentDB se actualizó en agosto de 2024. Si utiliza clústeres de Amazon DocumentDB con Seguridad de la capa de transporte (TLS) habilitada (la configuración predeterminada) y no ha rotado los certificados de servidor y aplicación cliente, se requieren los pasos siguientes para mitigar los problemas de conectividad entre la aplicación y los clústeres de Amazon DocumentDB.

- Paso 1: Descargar el nuevo certificado CA y actualizar la aplicación
- Paso 2: Actualizar el certificado de servidor

Los certificados CA y del servidor se actualizaron de conformidad con las prácticas recomendadas de mantenimiento y seguridad estándar para Amazon DocumentDB. Las aplicaciones cliente deben añadir los nuevos certificados CA a sus almacenes de confianza y las instancias de Amazon DocumentDB existentes deben actualizarse para utilizar los nuevos certificados CA antes de esta fecha de vencimiento.

Actualización de la aplicación y del clúster de Amazon DocumentDB

Siga los pasos indicados en esta sección para actualizar el grupo de certificados CA de la aplicación (Paso 1) y los certificados de servidor del clúster (Paso 2). Antes de aplicar los cambios a los entornos de producción, recomendamos encarecidamente probar estos pasos en un entorno de desarrollo o ensayo.

> **Note**
>
> Debe completar los pasos 1 y 2 en cada uno de los casos Región de AWS en los que tenga clústeres de Amazon DocumentDB.

Paso 1: Descargar el nuevo certificado CA y actualizar la aplicación

Descargue el nuevo certificado CA y actualice su aplicación de tal forma que use este nuevo certificado CA para crear las conexiones TLS a Amazon DocumentDB. Descargue el nuevo paquete de certificados CA desde https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem. Esta operación descarga un archivo llamado `global-bundle.pem`.

> **Note**
>
> Si accede al almacén de claves que incluye tanto el certificado CA antiguo (`rds-ca-2019-root.pem`) como los nuevos (`rds-ca-rsa2048-g1`, `rds-ca-rsa4096-g1`, `rds-ca-ecc384-g1`) compruebe que el almacén de claves tenga seleccionado `global-bundle`.

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

A continuación, actualice las aplicaciones de tal forma que utilicen el nuevo paquete de certificados. El nuevo paquete de CA contiene tanto el certificado CA anterior (rds-ca-2019) como los certificados CA nuevos (rds-ca-rsa2048-g1, rds-ca-rsa4096-g1, rds-ca-ecc384-g1). Al tener ambos certificados CA en el nuevo paquete de CA puede actualizar la aplicación y el clúster en dos pasos.

Para las aplicaciones de Java, se debe crear un nuevo almacén de confianza con el nuevo certificado CA. Para obtener instrucciones, consulte la pestaña de Java del tema Conexión con TLS habilitado.

Para comprobar que la aplicación utiliza el último grupo de certificados CA, consulte ¿Cómo puedo estar seguro de que estoy usando el paquete de CA más reciente?. Si ya está utilizando el último grupo de certificados CA en la aplicación, puede saltar al paso 2.

Para obtener ejemplos de uso de un grupo de CA con su aplicación, consulte Cifrado de datos en tránsito y Conexión con TLS habilitado.

> **Note**
>
> Actualmente, MongoDB Go Driver 1.2.1 sólo acepta un certificado de servidor de CA en `sslcertificateauthorityfile`. Consulte Conexión con TLS habilitado para conectarse a Amazon DocumentDB mediante Go cuando TLS esté habilitado.

Paso 2: Actualizar el certificado de servidor

Después de que la aplicación se haya actualizado con el fin de utilizar el nuevo paquete de CA, el siguiente paso consiste en actualizar el certificado de servidor al modificar cada instancia en un clúster de Amazon DocumentDB. Para modificar instancias para utilizar el nuevo certificado de servidor, consulte las instrucciones siguientes.

Amazon DocumentDB proporciona las siguientes CA para firmar el certificado del servidor de base de datos de una instancia de base de datos:

- rds-ca-ecc384-g1: utiliza una entidad de certificación con el algoritmo de clave privada RSA 384 y el algoritmo de firma SHA384. Esta CA admite la rotación automática de certificados de servidor. Solo es compatible con Amazon DocumentDB 4.0 y 5.0.
- rds-ca-rsa2048-g1: utiliza una autoridad de certificación con el algoritmo de clave privada RSA 2048 y el algoritmo de firma SHA256 en la mayoría de los casos. Regiones de AWS Esta CA admite la rotación automática de certificados de servidor.
- rds-ca-rsa4096-g1: utiliza una entidad de certificación con el algoritmo de clave privada RSA 4096 y el algoritmo de firma SHA384. Esta CA admite la rotación automática de certificados de servidor.

> **Note**
>
> Si utiliza el AWS CLI, puede ver la validez de las autoridades de certificación enumeradas en la sección anterior utilizando describe-certificates. https://docs.aws.amazon.com/cli/latest/reference/docdb/describe-certificates.html

Estos certificados CA se incluyen en el paquete de certificados regionales y globales. Al utilizar la CA rds-ca-rsa2048-g1, rds-ca-rsa4096-g1 o rds-ca-ecc384-g1 con una base de datos, Amazon DocumentDB administra el certificado del servidor de base de datos en la base de datos. Amazon

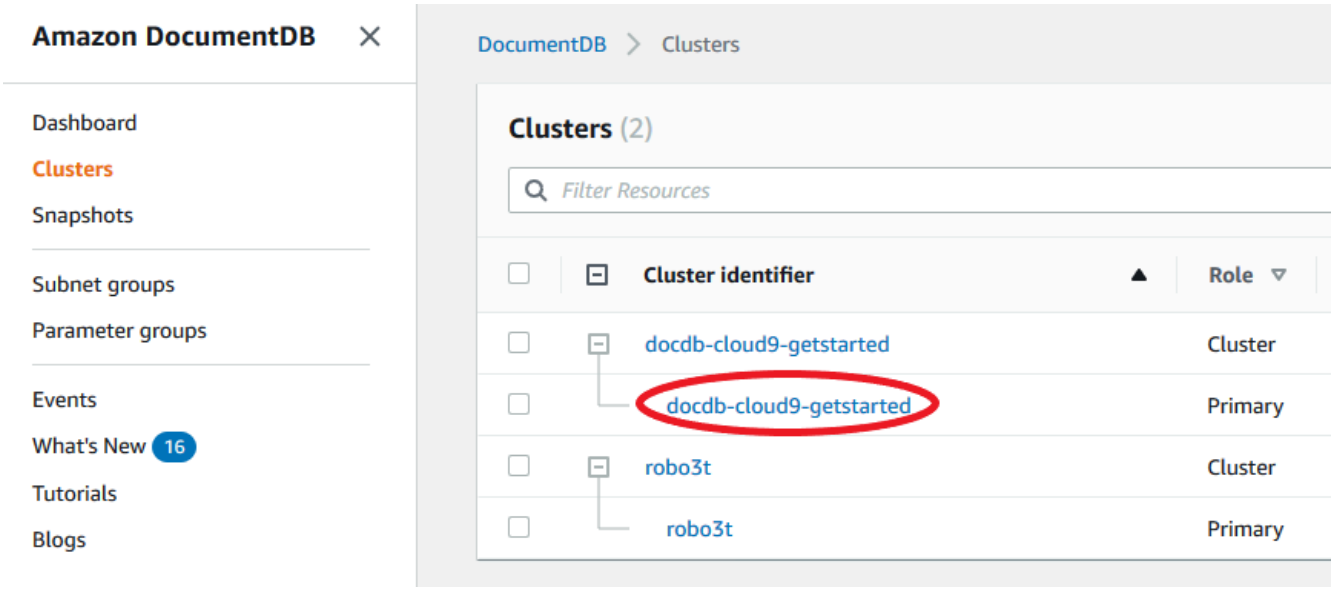
DocumentDB rota el certificado del servidor de base de datos de forma automática antes de que caduque.

> **Note**
>
> Amazon DocumentDB no requiere un reinicio para la rotación de certificados si el clúster se ejecuta en las siguientes versiones de parches del motor:
>
> - Amazon DocumentDB 3.6: 1.0.208662 o posterior
> - Amazon DocumentDB 4.0: 2.0.10179 o posterior
> - Amazon DocumentDB 5.0: 3.0.4780 o posterior
>
> Para determinar la versión de parche actual del motor de Amazon DocumentDB, ejecute el siguiente comando: `db.runCommand({getEngineVersion: 1})`.
> Antes de actualizar el certificado de servidor, asegúrese de haber completado Paso 1.

Using the Consola de administración de AWS

Complete los pasos siguientes para identificar y rotar el certificado de servidor antiguo para las instancias de Amazon DocumentDB existentes mediante la Consola de administración de AWS.

1. Inicie sesión en y abra la consola de Consola de administración de AWS Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En la lista de regiones de la esquina superior derecha de la pantalla, elija la región Región de AWS en la que residen sus clústeres.
3. En el panel de navegación en el lado izquierdo de la consola, en DAX, elija Clústeres.
4. Es posible que necesite identificar qué instancias siguen en el antiguo certificado de servidor (`rds-ca-2019`). Puede hacerlo en la columna Autoridad de certificación, que se encuentra en el extremo derecho de la tabla Clústeres.
5. En la tabla Clústeres, verá la columna Identificador del clúster en el extremo izquierdo. Tus instancias aparecen en clústeres, de forma similar a la siguiente captura de pantalla.



6. Marque la casilla de verificación situada a la izquierda de la instancia de su interés.
7. Elija Actions (Acciones) y después Modify (Modificar).
8. En Certificate authority (Entidad de certificación), seleccione el nuevo certificado de servidor (`rds-ca-rsa2048-g1`) para esta instancia.
9. Puede ver un resumen de los cambios en la página siguiente. Tenga en cuenta que se muestra una alerta adicional para recordarle que debe asegurarse de que su aplicación utilice el paquete más reciente de certificados CA antes de modificar la instancia, con el fin de evitar que se interrumpa la conectividad.
10. Puede optar por aplicar la modificación durante el próximo periodo de mantenimiento o bien aplicarla de inmediato. Si su intención es modificar el certificado del servidor inmediatamente, utilice la opción Apply immediately (Aplicar inmediatamente).
11. Seleccione Modify instance (Modificar instancia) para completar la actualización.

Using the AWS CLI

Complete los pasos siguientes para identificar y rotar el certificado de servidor antiguo para las instancias de Amazon DocumentDB existentes mediante la AWS CLI.

1. Para modificar las instancias de inmediato, ejecute el siguiente comando para cada instancia del clúster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa2048-g1 --apply-immediately
```

2. Para modificar las instancias de los clústeres de tal forma que usen el nuevo certificado CA durante el próximo periodo de mantenimiento del clúster, ejecute el siguiente comando para cada instancia del clúster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa2048-g1 --no-apply-immediately
```

Rotación automática de certificados del servidor

Amazon DocumentDB admite la rotación automática de certificados del servidor. El certificado del servidor es el certificado de entidad final emitido para cada instancia del clúster. A diferencia de los certificados CA raíz, los certificados del servidor tienen una validez corta (12 meses) y Amazon DocumentDB gestiona su rotación automáticamente sin que usted tenga que realizar ninguna acción. Amazon DocumentDB utiliza la misma CA raíz para esta rotación automática, por lo que no es necesario descargar un nuevo paquete de CA.

> **Important**
>
> Cuando se conecte a su clúster de Amazon DocumentDB, confíe en el paquete de CA raíz en lugar de confiar directamente en cada certificado de servidor. Esto evitará errores de conexión después de rotar el certificado del servidor. Consulte Conexión con TLS habilitado.

Amazon DocumentDB intenta rotar el certificado del servidor durante el período de mantenimiento que prefiera a la mitad de la vida del certificado del servidor. El nuevo certificado del servidor es válido durante 12 meses.

Utilice el comando `describe-db-engine-versions` e inspeccione el indicador `SupportsCertificateRotationWithoutRestart` para identificar si la versión del motor admite la rotación del certificado sin reinicio.

> **Note**
>
> Amazon DocumentDB admite la rotación de certificados del servidor sin reinicio si el clúster se ejecuta en las siguientes versiones de parches del motor:
>
> - Amazon DocumentDB 3.6: 1.0.208662 o posterior
> - Amazon DocumentDB 4.0: 2.0.10179 o posterior

- Amazon DocumentDB 5.0: 3.0.4780 o posterior

Para determinar la versión del parche actual del motor de Amazon DocumentDB, ejecute este comando: `db.runCommand({getEngineVersion: 1})`.
Si utiliza una versión anterior del parche del motor, Amazon DocumentDB rotará el certificado del servidor y programará un evento de reinicio de la base de datos en el período de mantenimiento que prefiera.

Resolución de problemas

Si tiene problemas para conectarse a su clúster como parte de la rotación de certificados, pruebe lo siguiente:

- Compruebe que sus clientes están utilizando el paquete de certificados más reciente. Consulte ¿Cómo puedo estar seguro de que estoy usando el paquete de CA más reciente?.
- Compruebe que las instancias están utilizando el certificado más reciente. Consulte ¿Cómo puedo saber cuáles de mis instancias de Amazon DocumentDB utilizan el certificado de old/new servidor?.
- Compruebe que la última entidad emisora de certificados está siendo utilizada por la aplicación. Algunos controladores, como Java y Go, requieren código adicional para importar varios certificados de un paquete de certificados al almacén de confianza. Para obtener más información sobre cómo conectarse a Amazon DocumentDB con TLS, consulte Conexión mediante programación a Amazon DocumentDB.
- Póngase en contacto con el soporte. Si tiene preguntas o problemas, póngase en contacto con Soporte.

Preguntas frecuentes

A continuación se presentan respuestas a algunas preguntas comunes acerca de los certificados TLS.

¿Qué hago si tengo preguntas o problemas?

Si tiene preguntas o problemas, póngase en contacto con Soporte.

¿Cómo sé si estoy usando TLS para conectarme a mi clúster de Amazon DocumentDB?

Para determinar si el clúster usa TLS, examine el parámetro `tls` del grupo de parámetros del clúster. Si el parámetro `tls` se encuentra establecido en `enabled`, está utilizando el certificado TLS para conectarse al clúster. Para obtener más información, consulte Administración de los grupos de parámetros de clúster de Amazon DocumentDB.

¿Por qué hay que actualizar los certificados CA y de servidor?

Los certificados CA y de servidor de Amazon DocumentDB se están actualizando de conformidad con las prácticas recomendadas de mantenimiento y seguridad estándar para Amazon DocumentDB.

¿Qué sucede si no realizo ninguna acción antes de la fecha de caducidad?

Si utiliza TLS para conectarse al clúster de Amazon DocumentDB con un certificado CA vencido, las aplicaciones que se conectan a través de TLS ya no podrán comunicarse más con el clúster de Amazon DocumentDB.

Amazon DocumentDB no rotará los certificados de la base de datos automáticamente antes de la fecha de caducidad. Debe actualizar las aplicaciones y los clústeres para utilizar los nuevos certificados CA antes o después de la fecha de caducidad.

¿Cómo puedo saber cuáles de mis instancias de Amazon DocumentDB utilizan el certificado de old/new servidor?

Para identificar las instancias de Amazon DocumentDB que aún utilizan el certificado de servidor anterior, puede utilizar Amazon DocumentDB Consola de administración de AWS o el. AWS CLI

Uso de Consola de administración de AWS

Identificación de las instancias de los clústeres que utilizan el certificado anterior

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En la lista de regiones de la esquina superior derecha de la pantalla, elija la región Región de AWS en la que residen sus instancias.
3. En el panel de navegación en el lado izquierdo de la consola, en DAX, elija Clústeres.

4. En la columna Autoridad de certificación (cerca del extremo derecho de la tabla), se muestran las instancias que todavía contienen el certificado de servidor antiguo (`rds-ca-2019`) y el certificado de servidor nuevo (`rds-ca-rsa2048-g1`).

Uso de AWS CLI

Para identificar las instancias de los clústeres que utilizan el certificado de servidor anterior, utilice el comando `describe-db-clusters` con lo siguiente.

```
aws docdb describe-db-instances \
    --filters Name=engine,Values=docdb \
    --query 'DBInstances[*].
{CertificateVersion:CACertificateIdentifier,InstanceID:DBInstanceIdentifier}'
```

¿Cómo puedo modificar instancias individuales de mi clúster de Amazon DocumentDB para actualizar el certificado de servidor?

Actualiza los certificados de servidor de todas las instancias de un clúster determinado al mismo tiempo. Para modificar las instancias del clúster, puede utilizar la consola o la AWS CLI.

> **Note**
>
> Antes de actualizar el certificado de servidor, asegúrese de haber completado el paso 1.

Uso de Consola de administración de AWS

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En la lista de regiones de la esquina superior derecha de la pantalla, elija la región Región de AWS en la que residen sus clústeres.
3. En el panel de navegación en el lado izquierdo de la consola, en DAX, elija Clústeres.
4. En la columna Autoridad de certificación (cerca del extremo derecho de la tabla), se muestran las instancias que todavía contienen el certificado de servidor antiguo (`rds-ca-2019`).
5. En la tabla Clústeres, en Identificador del clúster, seleccione una instancia para modificarla.
6. Elija Actions (Acciones) y después Modify (Modificar).

7. En Certificate authority (Entidad de certificación), seleccione el nuevo certificado de servidor (`rds-ca-rsa2048-g1`) para esta instancia.
8. Puede ver un resumen de los cambios en la página siguiente. Tenga en cuenta que se muestra una alerta adicional para recordarle que debe asegurarse de que su aplicación utilice el paquete más reciente de certificados CA antes de modificar la instancia, con el fin de evitar que se interrumpa la conectividad.
9. Puede optar por aplicar la modificación durante el próximo periodo de mantenimiento o bien aplicarla de inmediato.
10. Seleccione Modify instance (Modificar instancia) para completar la actualización.

Uso de AWS CLI

Complete los pasos siguientes para identificar y rotar el certificado de servidor antiguo para las instancias de Amazon DocumentDB existentes mediante la AWS CLI.

1. Para modificar las instancias de inmediato, ejecute el siguiente comando para cada instancia del clúster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa2048-g1 --apply-immediately
```

2. Para modificar las instancias de los clústeres de tal forma que usen el nuevo certificado CA durante el próximo periodo de mantenimiento del clúster, ejecute el siguiente comando para cada instancia del clúster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa2048-g1 --no-apply-immediately
```

¿Qué sucede si añado una nueva instancia a un clúster existente?

Todas las instancias nuevas que se creen utilizan el certificado de servidor antiguo y requieren conexiones TLS que usen el certificado CA antiguo. Cualquier instancia nueva de Amazon DocumentDB creada después del 25 de enero de 2024 utilizará de forma predeterminada el nuevo certificado rds-ca-rsa2048-g1.

¿Qué sucede si se sustituye una instancia o se produce una conmutación por error en el clúster?

Si se sustituye una instancia en el clúster, la nueva instancia que se crea continúa usando el mismo certificado de servidor que se estaba utilizando anteriormente. Se recomienda actualizar los certificados de servidor para todas las instancias al mismo tiempo. Si se produce una conmutación por error en el clúster, se usará el certificado de servidor del nuevo nodo principal.

Si no uso TLS para conectarme al clúster, ¿tengo que actualizar todas las instancias?

Recomendamos encarecidamente habilitar TLS. En caso de que no habilite TLS, le recomendamos que rote los certificados de las instancias de Amazon DocumentDB si planea usar TLS para conectarse a los clústeres en el futuro. Si no planea usar nunca TLS para conectarse a los clústeres de Amazon DocumentDB, no es necesaria ninguna acción.

Si no estoy usando TLS para conectarme a mi clúster pero tengo previsto hacerlo en el futuro, ¿qué tengo que hacer?

Si creó un clúster antes de enero de 2024, siga el paso 1 y el paso 2 de la sección anterior para asegurarse de que la aplicación utiliza la agrupación de CA actualizada y de que cada instancia de Amazon DocumentDB utiliza el certificado de servidor más reciente. Si crea un clúster después del 25 de enero de 2024, el clúster ya tendrá el certificado de servidor más reciente (rds-ca-rsa2048-g1). Para comprobar que la aplicación utiliza el grupo de CA más reciente, consulte Si no uso TLS para conectarme al clúster, ¿tengo que actualizar todas las instancias?

¿Puede prorrogarse el plazo más allá de agosto de 2024?

Si sus aplicaciones se conectan a través de TLS, la fecha límite no se podrá prorrogar.

¿Cómo puedo estar seguro de que estoy usando el paquete de CA más reciente?

Para comprobar que tiene el paquete más reciente, use el siguiente comando. Para ejecutar este comando, debe tener Java instalado y las herramientas de Java deben estar en la variable PATH de su intérprete de comandos. Para obtener más información, consulte Uso de Java.

macOS y Amazon Linux

```
keytool -printcert -v -file global-bundle.pem
```

Windows

```
keytool -printcert -v -file global-bundle.p7b
```

¿Por qué veo “RDS” en el nombre del paquete de CA?

Para determinadas funciones de administración, como la administración de certificados, Amazon DocumentDB utiliza tecnología operativa que se comparte con Amazon Relational Database Service (Amazon RDS).

¿Cuándo caduca el nuevo certificado?

El nuevo certificado del servidor caducará (por lo general) de la siguiente manera:

- rds-ca-rsa2048-g1: caduca en 2061
- rds-ca-rsa4096-g1: caduca en 2121
- rds-ca-ecc384-g1: caduca en 2121

¿Qué tipo de errores veré si no tomo medidas antes de que caduque el certificado?

Los mensajes de error variarán en función del controlador. En general, verá errores de validación de certificados que contienen la cadena “el certificado ha caducado”.

Si he aplicado el nuevo certificado de servidor, ¿puedo revertirlo al certificado de servidor antiguo?

Si necesitas revertir una instancia al certificado de servidor anterior, hazlo para todas las instancias del clúster. Puedes revertir el certificado de servidor de cada instancia de un clúster mediante el Consola de administración de AWS o el. AWS CLI

Uso de Consola de administración de AWS

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En la lista de regiones de la esquina superior derecha de la pantalla, elija la región Región de AWS en la que residen sus clústeres.
3. En el panel de navegación en el lado izquierdo de la consola, en DAX, elija Clústeres.

4. En la tabla Clústeres, en Identificador del clúster, seleccione una instancia para modificarla. Elija Actions (Acciones) y después Modify (Modificar).
5. En Certificate authority (Entidad de certificación), puede seleccionar el certificado de servidor antiguo (`rds-ca-2019`).
6. Elija Continue (Continuar) para ver un resumen de las modificaciones.
7. En la página que aparece, puede elegir programar las modificaciones para que se apliquen en el próximo periodo de mantenimiento o aplicar las modificaciones inmediatamente. Realice la selección y elija Modify instance (Modificar instancia).

> **Note**
>
> Si opta por aplicar las modificaciones inmediatamente, también se aplican los cambios de la cola de modificaciones pendientes. Si alguna de las modificaciones pendientes requiere un tiempo de inactividad, elegir esta opción puede provocar un tiempo de inactividad inesperado.

Uso de AWS CLI

```
aws docdb modify-db-instance --db-instance-identifier <db_instance_name> ca-
certificate-identifier rds-ca-2019 <--apply-immediately | --no-apply-immediately>
```

Si elige `--no-apply-immediately`, los cambios se aplicarán durante el próximo periodo de mantenimiento del clúster.

Si restauro a partir de una instantánea o de una restauración puntual, ¿tendrá el nuevo certificado de servidor?

Si restaura una instantánea o realiza una restauración puntual después de agosto de 2024, el nuevo clúster que se crea utilizará el nuevo certificado CA.

¿Qué sucede si tengo problemas para conectarme directamente a mi clúster de Amazon DocumentDB desde cualquier versión de Mac OS?

Mac OS ha actualizado los requisitos para certificados de confianza. Ahora los certificados de confianza deben tener una validez de 397 días o menos (consulte https://support.apple.com/en-us/HT211025).

> **Note**
>
> Esta restricción se observa en las versiones más recientes de Mac OS.

Los certificados de instancia de Amazon DocumentDB son válidos durante más de cuatro años, más que el máximo de Mac OS. Para conectarse directamente a un clúster de Amazon DocumentDB desde un equipo que ejecuta Mac OS, debe permitir certificados no válidos al crear la conexión TLS. En este caso, los certificados no válidos significan que el período de validez es superior a 397 días. Debe comprender los riesgos antes de permitir certificados no válidos al conectarse al clúster de Amazon DocumentDB.

Para conectarse a un clúster de Amazon DocumentDB desde Mac OS mediante el AWS CLI, utilice el parámetro. `tlsAllowInvalidCertificates`

```
mongo --tls --host <hostname> --username <username> --password <password> --port 27017
 --tlsAllowInvalidCertificates
```

Validación de la conformidad en Amazon DocumentDB

Auditores externos evalúan la seguridad y la conformidad de Amazon DocumentDB como parte de varios programas de conformidad de AWS , incluidos los siguientes:

- Controles del Sistema y Organizaciones (System and Organization Controls, SOC) 1, 2 y 3. Para obtener más información, consulte SOC.
- Programa Federal de Administración de Riesgos y Autorizaciones (FedRAMP). Para obtener más información, consulte Servicios de AWS en el ámbito del programa de conformidad.
- Estándar de Seguridad de Datos del Sector de las Tarjetas de Pago (PCI DSS, Payment Card Industry Data Security Standard). Para obtener más información, consulte PCI DSS.
- ISO 9001, 27001, 27017 y 27018. Para obtener más información, consulte Certificado ISO.
- Acuerdo para socio empresarial de la ley de portabilidad y responsabilidad de seguros médicos (HIPAA BAA). Para obtener más información, consulte Conformidad con HIPAA.

AWS proporciona una lista actualizada con frecuencia de AWS los servicios incluidos en el ámbito de programas de cumplimiento específicos en la sección AWS Servicios incluidos en el ámbito del programa de cumplimiento.

Third-party los informes de auditoría están disponibles para descargarlos mediante AWS Artifact. Para obtener más información, consulta Descarga de informes en AWS Artifact.

Para obtener más información sobre los programas AWS de cumplimiento, consulte Programas AWS de cumplimiento.

Su responsabilidad de conformidad al utilizar Amazon DocumentDB se determina en función de la sensibilidad de los datos, los objetivos de conformidad de la organización, así como de la legislación y los reglamentos aplicables. Si el uso de Amazon DocumentDB está sujeto al cumplimiento de normas como la HIPAA o la PCI, AWS proporciona recursos que le ayudarán a:

- AWS Recursos de cumplimiento: una colección de libros de trabajo y guías que pueden aplicarse a su sector y ubicación.
- Guías de inicio rápido sobre seguridad y cumplimiento: guías de implementación en las que se analizan las consideraciones arquitectónicas y se proporcionan los pasos para implementar entornos básicos centrados en la seguridad y el cumplimiento. AWS
- AWS Configuración: un servicio que evalúa en qué medida las configuraciones de sus recursos cumplen con las prácticas internas, las directrices del sector y las normativas.
- AWS Security Hub: una visión completa del estado de su seguridad interna AWS que le ayuda a comprobar el cumplimiento de los estándares y las mejores prácticas del sector de la seguridad.
- Documento técnico sobre la arquitectura para la seguridad y el cumplimiento de la HIPAA: documento técnico que describe cómo las empresas pueden utilizar para crear aplicaciones. AWS HIPAA-compliant

Resiliencia de Amazon DocumentDB

La infraestructura AWS global se basa en zonas de disponibilidad. Regiones de AWS Regiones de AWS proporcionan varias zonas de disponibilidad físicamente separadas y aisladas, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre zonas de disponibilidad sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Un clúster de base de Amazon DocumentDB solo se puede crear en una Amazon VPC que tenga un mínimo de dos subredes en al menos dos zonas de disponibilidad. Mediante la distribución de las

instancias del clúster en al menos dos zonas de disponibilidad, Amazon DocumentDB contribuye a garantizar que haya instancias disponibles en el clúster de base de datos en el caso improbable de que se produzca un error en una zona de disponibilidad. El volumen de un clúster de base de datos de Amazon DocumentDB siempre abarca tres zonas de disponibilidad. De este modo, se proporciona un almacenamiento duradero y con un menor riesgo de pérdida de datos.

Para obtener más información sobre las zonas Regiones de AWS de disponibilidad, consulte AWS Infraestructura global.

Además de la infraestructura AWS global, Amazon DocumentDB ofrece varias funciones que ayudan a satisfacer sus necesidades de respaldo y resiliencia de datos.

Fault-tolerant y almacenamiento con recuperación automática

Cada porción de 10 GB del volumen de almacenamiento se replica de seis maneras, en tres zonas de disponibilidad. Amazon DocumentDB utiliza un almacenamiento tolerante a errores que gestiona de forma transparente la pérdida de hasta dos copias de datos sin que ello afecte a la disponibilidad de escritura de la base de datos y de hasta tres copias sin que ello afecte a la disponibilidad de lectura. El almacenamiento de Amazon DocumentDB también se recupera automáticamente; los bloques de datos y los discos se escanean continuamente para detectar errores y se sustituyen automáticamente.

Copias de seguridad y restauración manuales

Amazon DocumentDB ofrece la capacidad de crear copias de seguridad completas de su clúster para retención y recuperación a largo plazo. Para obtener más información, consulte Copia de seguridad y restauración en Amazon DocumentDB.

Point-in-time recuperación

Point-in-time la recuperación ayuda a proteger los clústeres de Amazon DocumentDB de las operaciones de escritura o eliminación accidentales. Al habilitar la recuperación a un momento dado, ya no hay que preocuparse por crear, mantener o planificar copias de seguridad bajo demanda. Para obtener más información, consulte Restauración a un momento dado.

Seguridad de la infraestructura en Amazon DocumentDB

Como servicio administrado, Amazon DocumentDB está protegido por una seguridad de red AWS global. Para obtener información sobre los servicios de AWS seguridad y cómo AWS protege la infraestructura, consulte AWS Cloud Security. Para diseñar su AWS entorno utilizando las mejores

prácticas de seguridad de la infraestructura, consulte Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Utiliza las llamadas a la API AWS publicadas para acceder a Amazon DocumentDB a través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Conjuntos de cifrado con un secreto directo perfecto (PFS), como el DHE (efímero) o el ECDHE (efímero con curva Diffie-Hellman elíptica). Diffie-Hellman La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Puede llamar a estas operaciones de la API desde cualquier ubicación de red. Puede utilizar políticas de Amazon DocumentDB para controlar el acceso desde puntos de conexión específicos de Amazon Virtual Private Cloud (Amazon VPC) o VPC específicas. De hecho, esto aísla el acceso de red a un determinado recurso de Amazon DocumentDB únicamente de la VPC específica de la red. AWS

> **Note**
>
> Amazon DocumentDB no admite políticas de acceso basadas en recursos.

Puntos finales de API e interfaz de VPC de Amazon DocumentDB (AWS PrivateLink)

> **Note**
>
> Los clústeres elásticos de Amazon DocumentDB no admiten puntos de enlace de AWS PrivateLink VPC.

Puede establecer una conexión privada entre los puntos de conexión de VPC y la API de Amazon DocumentDB mediante la creación de un punto de conexión de VPC de interfaz. Los puntos finales de la interfaz funcionan con. AWS PrivateLink

Si bien los clústeres basados en instancias de Amazon DocumentDB no requieren una conexión de punto final de VPC desde la interfaz, AWS PrivateLink permiten acceder de forma privada a las

operaciones de la API de Amazon DocumentDB sin necesidad de una puerta de enlace a Internet, un dispositivo NAT, una conexión VPN o una conexión. Direct Connect Las instancias de Amazon DocumentDB de su VPC no necesitan direcciones IP públicas para comunicarse con los puntos de conexión de la API de Amazon DocumentDB para lanzar, modificar o terminar instancias de base de datos y clústeres de base de datos. Las instancias de Amazon DocumentDB tampoco necesitan direcciones IP públicas para utilizar ninguna de las operaciones de la API de Amazon DocumentDB disponibles. El tráfico entre la VPC y Amazon DocumentDB no sale de la red de Amazon.

Cada punto de conexión de la interfaz está representado por una o más interfaces de redes elásticas en las subredes. Para obtener más información, consulte Interfaces de red elásticas en la Guía del usuario de Amazon EC2.

Para obtener más información sobre los puntos de enlace de la VPC, consulte Cómo acceder a un punto de enlace de la VPC Servicio de AWS mediante una interfaz en la guía del usuario de Amazon Virtual Private Cloud ().AWS PrivateLink Para obtener más información sobre las operaciones de Amazon DocumentDB, consulte la referencia de la API de Amazon DocumentDB.

Temas

- Consideraciones para los puntos de conexión de VPC
- Disponibilidad por región
- Creación de un punto de conexión de VPC de interfaz para la API de Amazon DocumentDB
- Creación de una política de puntos de conexión de VPC para la API de Amazon DocumentDB

Consideraciones para los puntos de conexión de VPC

Antes de configurar un punto de conexión de VPC de tipo interfaz para los puntos de conexión de la API de Amazon DocumentDB, asegúrese de revisar los requisitos previos del puntos de conexión de interfaz en la Guía del usuario de nube privada virtual (AWS PrivateLink) de Amazon.

Todas las operaciones de API de Amazon DocumentDB pertinentes de la administración de recursos de Amazon DocumentDB están disponibles desde la VPC mediante el uso de AWS PrivateLink.

Las políticas de puntos de conexión de VPC son compatibles con los puntos de conexión de API de Amazon DocumentDB. De forma predeterminada, se permite el acceso completo a las operaciones de API de Amazon DocumentDB a través del punto de conexión. Para obtener más información, consulte Uso de políticas de punto de conexión para controlar el acceso a puntos de conexión de VPC en la Guía del usuario de Amazon Virtual Private Cloud (AWS PrivateLink).

Disponibilidad por región

La API de Amazon DocumentDB admite actualmente los siguientes puntos de enlace de VPC: Regiones de AWS

- Este de EE. UU. (Ohio)
- Este de EE. UU. (Norte de Virginia)
- Oeste de EE. UU. (Oregón)
- África (Ciudad del Cabo)
- Asia-Pacífico (Hong Kong)
- Asia-Pacífico (Mumbai)
- Asia-Pacífico (Hyderabad)
- Asia-Pacífico (Osaka)
- Asia-Pacífico (Seúl)
- Asia-Pacífico (Singapur)
- Asia-Pacífico (Sídney)
- Asia-Pacífico (Tokio)
- Canadá (centro)
- China (Pekín)
- China (Ningxia)
- Europa (Fráncfort)
- Europa (Irlanda)
- Europa (Londres)
- Europa (París)
- Europa (España)
- Europa (Milán)
- Medio Oriente (EAU)
- América del Sur (São Paulo)
- AWS GovCloud (US-East)
- AWS GovCloud (US-West)

Creación de un punto de conexión de VPC de interfaz para la API de Amazon DocumentDB

Puede crear un punto de enlace de VPC para la API de Amazon DocumentDB mediante la consola de Amazon VPC o el (). AWS Command Line Interface AWS CLI Para obtener más información, consulte Acceso y Servicio de AWS uso de un punto de enlace de VPC de interfaz en la guía del usuario de Amazon Virtual Private Cloud ()AWS PrivateLink.

Cree un punto de conexión de VPC para la API de Amazon DocumentDB mediante el uso del nombre del servicio `com.amazonaws.region.rds`.

Excepto Regiones de AWS en China, si habilita el DNS privado para el punto de enlace, puede realizar solicitudes de API a Amazon DocumentDB con el punto de enlace de la VPC utilizando su nombre de DNS predeterminado para, por ejemplo Regiones de AWS, rds.us-east-1.amazonaws.com. Para China (Pekín) y China (Ningxia) Regiones de AWS, puede realizar solicitudes de API con el punto de enlace de la VPC mediante rds-api.cn-north-1.amazonaws.com .cn y rds-api.cn-northwest-1.amazonaws.com .cn, respectivamente.

Para obtener más información, consulte Acceder a un punto de enlace de la VPC Servicio de AWS mediante una interfaz en la guía del usuario de Amazon Virtual Private Cloud ().AWS PrivateLink

Creación de una política de puntos de conexión de VPC para la API de Amazon DocumentDB

Puede asociar una política de punto de conexión al punto de conexión de VPC que controla el acceso a la API de Amazon DocumentDB. La política especifica la siguiente información:

- La entidad principal que puede realizar acciones.
- Las acciones que se pueden realizar.
- Los recursos en los que se pueden llevar a cabo las acciones.

Para obtener más información, consulte Uso de políticas de punto de conexión para controlar el acceso a puntos de conexión de VPC en la Guía del usuario de Amazon Virtual Private Cloud (AWS PrivateLink).

Ejemplo: política de punto de conexión de VPC para acciones de la API de Amazon DocumentDB

El siguiente es un ejemplo de una política de puntos finales para la API de Amazon DocumentDB. Cuando se asocia a un punto de conexión, esta política concede acceso a las acciones de la API de Amazon DocumentDB enumeradas para todas las entidades principales de todos los recursos.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": "*",
      "Effect": "Allow",
      "Action": [
        "rds:CreateDBInstance",
        "rds:ModifyDBInstance",
        "rds:CreateDBSnapshot"
      ],
      "Resource": "*"
    }
  ]
}
```

Ejemplo: política de puntos de conexión de la VPC que deniega todo acceso a una persona específica Cuenta de AWS

La siguiente política de punto final de la VPC niega a Cuenta de AWS 123456789012 todo el acceso a los recursos que utilizan el punto de enlace. La política permite todas las acciones de otros. Cuentas de AWS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
    }
```

```
  ]
}
```

Prácticas recomendadas de seguridad para Amazon DocumentDB

Amazon DocumentDB ofrece funciones de seguridad que incluyen el aislamiento de Amazon VPC, el cifrado en reposo AWS KMS, el cifrado en tránsito mediante TLS y un control de acceso detallado a través de IAM. Utilice las siguientes prácticas recomendadas para proteger los clústeres, las instancias y los datos de Amazon DocumentDB.

Utilice los roles AWS Identity and Access Management (IAM) para controlar el acceso a las operaciones de la API de Amazon DocumentDB. Esto incluye las operaciones que crean, modifican o eliminan recursos de Amazon DocumentDB, como clústeres, grupos de seguridad y grupos de parámetros. Cuando las aplicaciones u otros AWS servicios necesiten acceder a los recursos de Amazon DocumentDB, utilice una función de IAM para proporcionar credenciales temporales en lugar de almacenar las credenciales a largo plazo en su aplicación. Al crear funciones de IAM, aplique el principio del mínimo privilegio. Para obtener más información, consulte los roles de IAM y los escenarios comunes para los roles en la Guía del usuario de IAM.

- Imponga privilegios mínimos con control de acceso basado en roles.
- No utilice el usuario Cuenta de AWS raíz para administrar los recursos de Amazon DocumentDB. En su lugar, utilice AWS Identity and Access Management Identity Center para crear usuarios humanos con credenciales temporales. Para obtener más información, consulte los usuarios del IAM Identity Center en la Guía del usuario de IAM.
- Otorgue a cada identidad el conjunto mínimo de permisos necesarios para desempeñar sus funciones.
- Usa las políticas de IAM para administrar los permisos y sigue el principio del mínimo privilegio. Para más información, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.
- Si utilizas usuarios de IAM con credenciales de larga duración, rota esas credenciales con regularidad.

> **Warning**
>
> En este escenario, se requieren usuarios de IAM con acceso programático y credenciales de larga duración, lo que supone un riesgo de seguridad. Para ayudar a mitigar este riesgo, le recomendamos que brinde a estos usuarios únicamente los permisos que

necesitan para realizar la tarea y que los elimine cuando ya no los necesiten. Las claves de acceso se pueden actualizar si es necesario. Para más información consulte Actualización de las claves de acceso en la Guía de usuario de IAM.

- Configure Secrets Manager para rotar automáticamente los secretos de Amazon DocumentDB. Para obtener más información, consulte Cómo rotar sus secretos y Cómo rotar los secretos para Amazon DocumentDB en la Guía del AWS Secrets Manager usuario.
- Utilice seguridad de la capa de transporte (TLS) y el cifrado en reposo para cifrar los datos.

Auditoría de eventos de Amazon DocumentDB

Con Amazon DocumentDB (compatible con MongoDB), puede auditar eventos que se realizaron en su clúster. Los intentos de autenticación correctos e incorrectos, la eliminación de una colección en una base de datos o la creación de un índice son algunos ejemplos de eventos registrados. De forma predeterminada, la auditoría está deshabilitada en Amazon DocumentDB y, para utilizar esta característica, es necesario suscribirse.

Cuando la auditoría está habilitada, Amazon DocumentDB registra en Amazon Logs los eventos del lenguaje de definición de datos (DDL), el lenguaje de manipulación de datos (DML), la autenticación, la autorización y la administración de usuarios. CloudWatch Cuando la auditoría está habilitada, Amazon DocumentDB exporta los registros de auditoría del clúster (documentos JSON) a Amazon Logs. CloudWatch Puede usar Amazon CloudWatch Logs para analizar, supervisar y archivar sus eventos de auditoría de Amazon DocumentDB.

Si bien Amazon DocumentDB no cobra ningún coste adicional por permitir la auditoría, se le cobran tarifas estándar por el uso de CloudWatch los registros. Para obtener información sobre CloudWatch los precios de Logs, consulte los CloudWatch precios de Amazon.

La función de auditoría de Amazon DocumentDB difiere claramente del uso de los recursos del servicio con la que se monitoriza. AWS CloudTrail CloudTrail registra las operaciones que se realizan con AWS Command Line Interface (AWS CLI) o Consola de administración de AWS en recursos como clústeres, instancias, grupos de parámetros e instantáneas. La auditoría de los recursos CloudTrail está activada de forma predeterminada y no se puede deshabilitar. La característica de auditoría de Amazon DocumentDB es opcional. Registra las operaciones que tienen lugar dentro del clúster en los distintos objetos, como, por ejemplo, bases de datos, colecciones, índices y usuarios.

Temas

- Eventos admitidos

- Habilitación de auditorías
- Deshabilitación de auditorías
- Acceso a los eventos de auditoría
- Filtrado de eventos de auditoría de DML

Eventos admitidos

La auditoría de Amazon DocumentDB admite las siguientes categorías de eventos:

- Lenguaje de definición de datos (DDL): incluye las operaciones de administración de bases de datos, las conexiones, la administración de usuarios y la autorización.
- Eventos de lectura del lenguaje de manipulación de datos (lecturas DML): incluye `find()` y los distintos operadores de agregación, operadores aritméticos, operadores booleanos y otros operadores de consulta de lectura.
- Eventos de escritura del lenguaje de manipulación de datos (escrituras en DML): incluye operadores `insert(), update(), delete(),` y `bulkWrite()`

Los tipos de eventos son los siguientes.

Tipo de evento	Categoría	Description (Descripción)
`authCheck`	Autorización	Código de resultado 0: éxito
		Código de resultado 13: intentos no autorizados de realizar una operación.
`authenticate`	Connection	Intentos de autenticación correctos e incorrectos en una nueva conexión.

Tipo de evento	Categoría	Description (Descripción)
`auditConfigure`	DDL	Auditoría de la configuración de filtrado.
`createDatabase`	DDL	Creación de una nueva base de datos.
`createCollection`	DDL	Creación de una nueva colección en una base de datos.
`createIndex`	DDL	Creación de un nuevo índice en una colección.
`dropCollection`	DDL	Eliminación de una nueva colección en una base de datos.
`dropDatabase`	DDL	Eliminación de una base de datos.
`dropIndex`	DDL	Eliminación de un índice en una colección.
`modifyChangeStreams`	DDL	Se creó el flujo de cambios.
`renameCollection`	DDL	Renombrar una nueva colección en una base de datos.
`createRole`	Administración de roles	Creación de un rol.

Tipo de evento	Categoría	Description (Descripción)
dropAllRolesFromDatabase	Administración de roles	Eliminación de todos los roles en una base de datos.
dropRole	Administración de roles	Descartando un rol.
grantPrivilegesToRole	Administración de roles	Concediendo privilegios a un rol.
grantRolesToRole	Administración de roles	Concediendo roles a un rol definido por el usuario.
revokePrivilegesFromRole	Administración de roles	Revocando los privilegios de un rol.
revokeRolesFromRole	Administración de roles	Revocando roles de un rol definido por el usuario.
updateRole	Administración de roles	Actualización de un rol.
createUser	Administración de usuarios	Creación de un nuevo usuario.
dropAllUsersFromDatabase	Administración de usuarios	Eliminación de todos los usuarios en una base de datos.
dropUser	Administración de usuarios	Eliminación de un usuario existente.
grantRolesToUser	Administración de usuarios	Concesión de roles a un usuario.

Tipo de evento	Categoría	Description (Descripción)
`revokeRolesFromUser`	Administración de usuarios	Revocando roles de un usuario.
`updateUser`	UserManagement	Actualización de un usuario existente.
`insert`	Escritura DML	Inserta uno o varios documentos en una colección.
`delete`	Escritura DML	Elimina uno o varios documentos de una colección.
`update`	Escritura DML	Modifica uno o varios documentos existentes en una colección.
`bulkWrite`	Escritura DML	Realiza múltiples operaciones de escritura con controles para determinar el orden de ejecución.
`setAuditConfig`	Escritura DML	Establezca un filtro nuevo para la auditoría de DML.
`count`	Lectura de DML	Devuelve el recuento de documentos que coincidirían con una consulta find() de la colección o vista.

Tipo de evento	Categoría	Description (Descripción)
`countDocuments`	Lectura de DML	Devuelve el recuento de documentos que coinciden con una consulta para una colección o vista.
`find`	Lectura de DML	Selecciona los documentos de una colección o vista y devuelve el cursor a los documentos seleccionados.
`getAuditConfig`	Lectura de DML	Recupere el filtro actual para la auditoría de DML.
`findAndModify`	Lectura de DML y escritura de DML	Modifica y devuelve un único documento.
`findOneAndDelete`	Lectura de DML y escritura de DML	Elimina un solo documento en función de los criterios de filtro y clasificación y devuelve el documento eliminado.
`findOneAndReplace`	Lectura de DML y escritura de DML	Sustituye un único documento en función del filtro especificado.
`findOneAndUpdate`	Lectura de DML y escritura de DML	Actualiza un único documento en función de los criterios de filtrado y clasificación.

Tipo de evento	Categoría	Description (Descripción)
`aggregate`	Lectura de DML y escritura de DML	Admite las API en la canalización de agregación.
`distinct`	Lectura de DML	Busca los valores distintos de un campo especificado en una sola colección o vista y devuelve los resultados en una matriz.

> **Note**
>
> Los valores del campo de parámetros del documento de eventos de DML tienen un límite de tamaño de 1 KB. Amazon DocumentDB trunca el valor si supera 1 KB.

> **Note**
>
> Los eventos de eliminación de TTL no se auditan en este momento.

Habilitación de auditorías

Para habilitar la auditoría en un clúster, hay que seguir un proceso de dos pasos. Asegúrese de completar ambos pasos o los registros de auditoría no se enviarán a CloudWatch Logs.

Paso 1. Habilitación del parámetro de clúster audit_logs

Para habilitar la auditoría, debe modificar el parámetro `audit_logs` en el grupo de parámetros. `audit_logs` es una lista delimitada por comas de los eventos que se deben registrar. Los eventos se deben especificar en minúsculas y no debe haber espacios en blanco entre los elementos de la lista.

Puede especificar los siguientes valores para el grupo de parámetro:

Valor	Description (Descripción)	
`ddl`	Si se establece esto, se habilitará la auditoría de eventos de DDL como CreateDatabase, DropDatabase, CreateCollection, DropCollection, CreateIndex, DropIndex, AuthCheck, authenticate, CreateUser, dropUser, grant, revoke, updateUser y drop RolesToUser RolesFromUser AllUsersFromDatabase	
`dml_read`	Cuando configura esto, se habilita la auditoría de eventos de lectura de DML como find, sort count, distinct, group, projecta, unwind, geoNear, geoIntersects, geoWithin y otros operadores de consulta de lectura de MongoDB.	

Valor	Description (Descripción)	
`dml_write`	Al configurarlo, se habilitará la auditoría de eventos de escritura de DML como insert(), update(), delete() y bulkWrite()	
`all`	Al configurarlo, se habilitará la auditoría de los eventos de la base de datos, como las consultas de lectura y escritura, las acciones de la base de datos y las acciones del administrador.	
`none`	Si lo configura, se deshabilitará la auditoría	

Valor	Description (Descripción)	
`enabled` (heredado)	Se trata de una configuración de parámetros antigua que equivale a “ddl”. Esta configuración permitirá la auditoría de eventos de DDL como CreateDatabase, DropDatabase, CreateCollection, DropCollection, CreateIndex, DropIndex, AuthCheck, authenticate, CreateUser, DropUserRolesToUser, grant, revokeRolesFromUser, updateUser y dropAllUsersFromDatabase. No se recomienda utilizar este ajuste porque se trata de un ajuste heredado.	

Valor	Description (Descripción)	
`disabled` (heredado)	Se trata de una configuración de parámetros antigua que equivale a “none”. No se recomienda utilizar este ajuste porque se trata de un ajuste heredado.	

> **Note**
>
> El valor predeterminado del parámetro de clúster audit_logs es none (“`disabled`” heredado).

También puede utilizar los valores mencionados anteriormente en combinaciones.

Valor	Description (Descripción)	
`ddl, dml_read`	Si lo configura, se habilitará la audición de los eventos DDL y los eventos de lectura de DML.	
`ddl, dml_write`	Si lo configura, se habilitará la audición de los eventos DDL y la lectura de DML.	

Valor	Description (Descripción)	
`dml_read, dml_write`	Si lo configura, se habilitará la audición de los eventos DDL	

> **Note**
>
> No es posible modificar un grupo de parámetros predeterminado.

Para obtener más información, consulte los siguientes temas:

- Creación de grupos de parámetros de clúster de Amazon DocumentDB

  Después de crear un grupo de parámetros, para modificarlo, debe cambiar el valor del parámetro `audit_logs` a `all`.
- Modificación de grupos de parámetros de clúster de Amazon DocumentDB

Paso 2. Habilita la exportación de Amazon Logs CloudWatch

Si el valor del parámetro de `audit_logs` clúster es`enabled`,`ddl`, o `dml_read`dml_write, también debe habilitar Amazon DocumentDB para exportar los registros a Amazon CloudWatch. Si omite alguno de estos pasos, los registros de auditoría no se enviarán a. CloudWatch

Al crear un clúster, realizar una restauración puntual o restaurar una instantánea, puede habilitar los CloudWatch registros siguiendo estos pasos.

Using the Consola de administración de AWS

Para permitir que Amazon DocumentDB exporte los registros al CloudWatch uso de la consola, consulte los siguientes temas:

- Al crear un clúster: en Crear un clúster y una instancia principal mediante el Consola de administración de AWS, consulte Creación de un clúster: configuraciones adicionales (paso 5, Exportaciones de registros)

- Al modificar un clúster existente — Modificación de un clúster de Amazon DocumentDB
- Al realizar una restauración de una instantánea de un clúster — Restauración de una instantánea del clúster
- Cuando se realiza una restauración a un momento dado — Restauración a un momento dado

Using the AWS CLI

Para habilitar los registros de auditoría al crear un nuevo clúster

El siguiente código crea el clúster `sample-cluster` y habilita los registros CloudWatch de auditoría.

Example

Para 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
```

Para 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
```

Para habilitar los registros de auditoría al modificar un clúster existente

El código siguiente modifica el clúster `sample-cluster` y habilita los registros de CloudWatch auditoría.

Example

Para Linux, macOS o Unix:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

Para Windows:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

El resultado de estas operaciones será similar al que se indica a continuación (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"
    }
}
```

Deshabilitación de auditorías

Para deshabilitar la auditoría, deshabilite la exportación de CloudWatch registros y deshabilite el parámetro. `audit_logs`

Inhabilitar CloudWatch la exportación de registros

Puede deshabilitar la exportación de registros de auditoría mediante el Consola de administración de AWS o el AWS CLI.

Using the Consola de administración de AWS

El siguiente procedimiento utiliza el Consola de administración de AWS para deshabilitar la exportación de registros de Amazon DocumentDB a CloudWatch.

Para deshabilitar los registros de auditoría

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Clusters (Clústeres). A continuación, elija el botón situado a la izquierda del nombre del clúster para el que desea deshabilitar la exportación de registros.
3. Elija Actions (Acciones) y después Modify (Modificar).

4. Desplácese hacia abajo hasta la sección Log exports (Exportaciones de registros) y, a continuación, elija Disabled (Deshabilitado).
5. Elija Continuar.
6. Revise los cambios y, a continuación, elija cuándo desea que se aplique este cambio en su clúster.

   - Apply during the next scheduled maintenance window (Aplicar durante el siguiente periodo de mantenimiento programado)
   - Apply immediately (Aplicar inmediatamente)
7. Elija Modificar clúster.

Using the AWS CLI

El código siguiente modifica el clúster `sample-cluster` e inhabilita CloudWatch los registros de auditoría.

Example

Para Linux, macOS o Unix:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit"]}'
```

Para Windows:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit"]}'
```

La salida de esta operación será similar a lo que se indica a continuación (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"
            }
        ]
    }
}
```

Deshabilitación del parámetro audit_logs

Para deshabilitar el parámetro `audit_logs` del clúster, puede modificar el clúster para que utilice un grupo de parámetros donde el valor del parámetro `audit_logs` esté `disabled`. O bien, puede modificar el valor del parámetro `audit_logs` en el grupo de parámetros del clúster para que esté `disabled`.

Para obtener más información, consulte los temas siguientes:

- Modificación de un clúster de Amazon DocumentDB
- Modificación de grupos de parámetros de clúster de Amazon DocumentDB

Acceso a los eventos de auditoría

Sigue los pasos siguientes para acceder a tus eventos de auditoría en Amazon. CloudWatch

1. Abre la CloudWatch consola en https://console.aws.amazon.com/cloudwatch/.
2. Asegúrese de que se encuentra en la misma región que el clúster de Amazon DocumentDB.
3. En el panel de navegación, elija Logs (Registros).
4. Para buscar los registros de auditoría del clúster, busque y elija en la lista **`/aws/docdb/`*`yourClusterName`*`/audit`**.

   Los eventos de auditoría de cada una de las instancias están disponibles debajo del nombre de instancia correspondiente.

Filtrado de eventos de auditoría de DML

Introducción al filtrado de auditoría de DML

Los eventos de auditoría de DML se pueden filtrar antes de escribirlos en Amazon CloudWatch. Para utilizar esta característica, el registro de auditoría y el registro de DML deben estar habilitados. Amazon DocumentDB admite el filtrado en `atype`, `command`, `user`, `namespace` y `auditAuthorizationSuccess`.

> **Note**
>
> Los eventos DDL no se filtran.

Para habilitar el filtrado de auditoría en cualquier momento, especifique el filtro de auditoría mediante los parámetros `setAuditConfig`, `filter` y `auditAuthorizationSuccess` de la operación `db.adminCommand( { command } )`:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
```

```
      filter:
         {
            //filter conditions
         },
      auditAuthorizationSuccess: true | false
   }
)
```

También puede recuperar la configuración del filtro de auditoría mediante la ejecución del siguiente comando:

```
db.admin.runCommand( { getAuditConfig: 1})
```

Requisitos de seguridad

Solo las bases de datos users/roles con acciones privilegiadas `auditConfigure` pueden ejecutar los comandos anteriores `admindb` al configurar o enumerar los filtros de auditoría de DML. Puede utilizar uno de los roles integrados de [`clusterAdmin`, `hostManager`, `root`] o crear roles personalizados con privilegios de `auditConfigure`. A continuación, se muestra un ejemplo del uso de los roles existentes con el privilegio de `auditConfigure` y un ejemplo con los roles personalizados.

Usuario con rol integrado:

```
use admin
db.createUser(
  {
    user: "myClusterAdmin",
    pwd: "password123",
    roles: [ { role: "clusterAdmin", db: "admin" } ]
  }
)
```

Usuario con roles personalizados:

```
use admin
db.createRole(
   {
     role: "myRole",
     privileges: [
       { resource: { cluster: true }, actions: [ "auditConfigure" ] }
```

```
     ],
     roles: []
   }
)
db.createUser(
  {
    user: "myUser",
    pwd: "myPassword",
    roles: [ { role: "myRole", db: "admin" } ]
  }
)
```

Filtrado de casos de uso

Ejemplo: filtrado de eventos por comandos

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
        "$and": [
         {
            "param.command":
               {
                  $in: [ "find","count", "insert", "delete", "update",
 "findandmodify" ]
               }
         }
         ]
      },
      auditAuthorizationSuccess: true
   }
)
```

Ejemplo: filtrado de eventos por nombre de usuario

En este ejemplo, solo se registrará el usuario “myUser”:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
```

```
         {
            "param.user":
               {
                  $in: [ "myUser" ]
               }
         }
         ]},
      auditAuthorizationSuccess: true})
```

Ejemplo: filtrado mediante **`atype`**

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {atype: "authCheck"},
      auditAuthorizationSuccess: true
   })
```

> **Note**
>
> Todos los registros de DML tienen `authCheck` como `atype`. Solo DDL tiene un `atype` diferente. Si introduce un valor distinto de `authCheck` en`filter`, no se generará un inicio de sesión en DML. CloudWatch

Ejemplo: filtrado mediante varios filtros unidos por operadores

```
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**
>
> En el nivel superior, solo se admiten `$and`, `$or` y `$nor`. Los demás operadores no se admiten y provocarán un error.

Ejemplo: filtrado mediante eventos por **`auditAuthorizationSuccess`**

En este filtro, no se registrarán todos los comandos que hayan pasado la autorización correctamente:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {},
      auditAuthorizationSuccess: false
   }
)
```

Ejemplo: filtrado con condiciones **`$in`** y **`$nin`**

Si se utiliza `$in` y `$nin`, el comando no se registrará, ya que habrá un “y” implícito entre las condiciones. En este ejemplo, regex bloqueará el comando `find` para que no se registre nada:

```
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})
```

Ejemplo: filtrado mediante **`namespace`**

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            "param.ns":
               {
                  $in: [ "test.foo" ]
               }
         }
         ]},
      auditAuthorizationSuccess: true})
```

Ejemplo: restablecimiento del filtro predeterminado

Si se restablece el valor predeterminado, se registrarán todos los eventos de auditoría de DML. Para restablecer el filtrado al valor predeterminado, ejecute el siguiente comando:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {},
      auditAuthorizationSuccess: true
   }
)
```

Amazon VPC y Amazon DocumentDB

Amazon Virtual Private Cloud (Amazon VPC) le permite lanzar AWS recursos, como las instancias de Amazon DocumentDB, en una nube privada virtual (VPC).

Cuando utilice una VPC, puede controlar todos los aspectos del entorno de red virtual. Puede elegir su propio rango de direcciones IP, crear subredes y configurar listas de enrutamiento y control de acceso. Es posible ejecutar el clúster en una VPC sin costo adicional.

Las cuentas tienen una VPC predeterminada. Todos los clústeres nuevos se crean en la VPC predeterminada, a menos que se especifique lo contrario.

Temas

- Clústeres de DocumentDB en una VPC
- Acceso a un clúster de Amazon DocumentDB en una VPC
- Cree una IPv4-only VPC para usarla con un clúster de DocumentDB
- Creación de una VPC de pila doble para utilizarla con un clúster de DocumentDB

A continuación, encontrará una discusión acerca de la funcionalidad de la VPC relevante para los clústeres de Amazon DocumentDB. Para obtener más información sobre Amazon VPC, consulte la Guía del usuario de Amazon VPC.

Clústeres de DocumentDB en una VPC

Su clúster de Amazon DocumentDB es una nube privada virtual (VPC). Una VPC es una red virtual que está aislada de forma lógica de otras redes virtuales del. Nube de AWS Amazon VPC le permite lanzar AWS recursos, como un clúster de Amazon DocumentDB o una instancia de Amazon EC2, en una VPC. La VPC puede ser una VPC predeterminada que viene con la cuenta o una que se haya creado en ella. Todas las VPC están asociadas a su cuenta. AWS

La VPC predeterminada tiene tres subredes que se pueden utilizar para aislar recursos dentro de la VPC. La VPC predeterminada también tiene una puerta de enlace de Internet que se puede utilizar para proporcionar acceso a los recursos situados dentro de la VPC desde fuera de la VPC.

Para ver las situaciones relacionadas con los clústeres de Amazon DocumentDB dentro y fuera de una VPC, consulte Acceso a un clúster de Amazon DocumentDB en una VPC.

Temas

- Trabajo con clústeres en una VPC
- Uso de grupos de subredes
- Direcciones IP de Amazon DocumentDB
- Creación de un clúster en una VPC

Trabajo con clústeres en una VPC

A continuación se ofrecen algunos consejos para trabajar con un clúster en una VPC:

- La VPC debe tener dos subredes como mínimo. Estas subredes deben estar en dos zonas de disponibilidad diferentes en el Región de AWS lugar donde desea implementar el clúster. Una subred es un segmento del rango de direcciones IP de una VPC que puede especificar y que le permite agrupar clústeres según sus necesidades operativas y de seguridad.
- Si desea que un clúster de la VPC sea accesible públicamente, asegúrese de activar los atributos `DNS hostnames` y `DNS resolution` de la VPC.
- La VPC debe tener un grupo de subredes que haya creado. Para crear un grupo de subredes, especifique las subredes que ha creado. Amazon DocumentDB elige una subred y una dirección IP dentro de esa subred para asociarla con la instancia principal de su clúster. La instancia principal utiliza la zona de disponibilidad que contiene la subred.
- La VPC debe tener un grupo de seguridad de VPC que permita el acceso al clúster.

  Para obtener más información, consulte Acceso a un clúster de Amazon DocumentDB en una VPC.
- Los bloques de CIDR de cada una de las subredes deben ser lo suficientemente grandes como para acomodar direcciones IP de repuesto para que Amazon DocumentDB las use durante las actividades de mantenimiento, incluyendo la conmutación por error y el escalado de recursos de computación. Por ejemplo, un rango como 10.0.0. 0/24 y 10.0.1. 0/24 suele ser lo suficientemente grande.
- Una VPC puede tener un atributo de `instance tenancy` de `default` o `dedicated`. Todas las VPC predeterminadas tienen el atributo de `instance tenancy` definido como predeterminado, y una VPC predeterminada puede admitir cualquier clase de instancia.

  Si opta por tener el clúster en una VPC dedicada cuyo atributo de `instance tenancy` está establecido en `dedicated`, la clase de instancia del clúster debe corresponder a uno de los tipos aprobados de instancia dedicada de Amazon EC2. Por ejemplo, la instancia de EC2 dedicada `r5.large` corresponde a la clase de instancia `db.r5.large`. Para obtener información acerca

de la tenencia de instancias en una VPC, consulte Instancias de Amazon EC2 en la Guía del usuario de Amazon Elastic Compute Cloud.

Para obtener más información acerca de los tipos de instancias que puede haber en una instancia dedicada, consulte Instancias dedicadas de Amazon EC2 en la página de precios de Amazon EC2.

> **Note**
>
> Cuando establece el atributo de `instance tenancy` en `dedicated` para un clúster, no garantiza que el clúster se ejecute en un host dedicado.

Uso de grupos de subredes

Las subredes son segmentos del rango de direcciones IP de una VPC que se definen para agrupar recursos de acuerdo con las necesidades operativas y de seguridad. Los grupos de subredes son una colección de subredes (normalmente privadas) que crea en una VPC y que después asigna a los clústeres. Al usar un grupo de subredes, puede especificar una VPC concreta al crear clústeres mediante la API AWS CLI o la API de Amazon DocumentDB. Si utiliza la consola, solo puede elegir la VPC y los grupos de subredes que desea utilizar.

Cada grupo de subredes debe tener subredes como mínimo en dos zonas de disponibilidad de una Región de AWS determinada. Cuando crea un clúster en una VPC, debe elegir un grupo de subredes. Del grupo de subredes, Amazon DocumentDB elige una subred y una dirección IP dentro de esa subred para asociarla con la instancia principal de su clúster. La base de datos utiliza la zona de disponibilidad que contiene la subred. DocumentDB siempre asigna una dirección IP desde una subred que tiene espacio de dirección IP libre.

Las subredes en un grupo de subredes son públicas o privadas, en función de la configuración que establezca para sus listas de control de acceso de la red (ACL de la red) y tablas de enrutamiento. Para que un clúster sea accesible públicamente, todas las subredes del grupo de subredes deben ser públicas. Si una subred asociada a un clúster de acceso público cambia de pública a privada, eso puede afectar la disponibilidad del clúster.

Para crear un grupo de subredes que admita el modo de pila doble, asegúrese de que cada subred que agregue al grupo de subredes tenga un bloque de CIDR de protocolo de Internet versión 6 (IPv6) asociado. Para obtener más información, consulte Direcciones IP de Amazon DocumentDB y Compatibilidad con IPv6 para su VPC en la Guía del usuario de la Amazon Virtual Private Cloud.

Cuando Amazon DocumentDB crea un clúster en una VPC, asigna una interfaz de red al clúster con la dirección IP seleccionada del grupo de subredes. Utilice siempre el nombre del sistema de nombres de dominio (DNS) para conectarse a su clúster, ya que la dirección IP subyacente cambia durante la conmutación por error.

> **Note**
>
> Para cada clúster que ejecute en una VPC, asegúrese de reservar al menos una dirección en cada subred del grupo de subredes para que la utilice Amazon DocumentDB para las acciones de recuperación.

Subredes compartidas

Puede crear un clúster en una VPC compartida.

Algunas consideraciones a tener en cuenta al utilizar las VPC compartidas:

- Puede mover un clúster de una subred de VPC compartida a una subred de VPC no compartida y viceversa.
- Los participantes de una VPC compartida deben crear un grupo de seguridad en la VPC que les permita crear un clúster.
- Los propietarios y los participantes de una VPC compartida pueden acceder a la base de datos mediante consultas en DocumentDB. Sin embargo, solo el creador de un recurso puede realizar llamadas a la API en el recurso.

Direcciones IP de Amazon DocumentDB

Las direcciones IP permiten que los recursos de la VPC se comuniquen entre sí y con otros recursos a través de Internet. Amazon DocumentDB admite los protocolos de direccionamiento IPv4 e IPv6. De forma predeterminada, Amazon DocumentDB y Amazon VPC utilizan el protocolo de direccionamiento IPv4. No puedes desactivar este comportamiento. Al crear una VPC, debe especificar un bloque de CIDR IPv4 (un intervalo de direcciones IPv4 privadas). De manera opcional, puede asignar un bloque de CIDR IPv6 a su VPC y sus subredes y asignar direcciones IPv6 de dicho bloque a clústeres de su subred

> **Note**
>
> Dual-stack el modo (direccionamiento IPv6) solo se admite en las versiones 4.0 y 5.0 de Amazon DocumentDB.

La compatibilidad con el protocolo IPv6 amplía el número de direcciones IP admitidas. Al utilizar el protocolo IPv6, se asegura de tener suficientes direcciones disponibles para el futuro crecimiento de Internet. Los recursos de DocumentDB nuevos y existentes pueden utilizar direcciones IPv4 e IPv6 dentro de su VPC. Configurar, proteger y traducir el tráfico de red entre los dos protocolos utilizados en diferentes partes de una aplicación puede provocar sobrecarga operativa. Puede estandarizar el protocolo IPv6 para los recursos de Amazon DocumentDB para simplificar la configuración de la red.

Temas

- Direcciones IPv4
- Direcciones IPv6
- Dual-stack modo

Direcciones IPv4

Al crear una VPC, debe especificar un rango de direcciones IPv4 para la VPC como bloque de CIDR como `10.0.0.0/16`. Un grupo de subredes define el rango de direcciones IP de este bloque de CIDR que puede utilizar un clúster. Esta dirección IP puede ser privada o pública.

Una dirección IPv4 privada es una dirección IP a la que no se puede obtener acceso desde Internet. Se pueden usar direcciones IPv4 privadas para la comunicación entre el clúster y otros recursos, como instancias de Amazon EC2, en la misma VPC. Cada clúster tiene una dirección IP privada para la comunicación en la VPC.

Una dirección IP pública es una dirección IPv4 a la que se puede acceder desde Internet. No se permiten las direcciones IP públicas en el clúster de DocumentDB. Cualquier dirección IP pública debe ser resuelta por la puerta de enlace de Internet y por EC2 en la subred pública.

Para ver cómo crear una VPC con solo direcciones IPv4 privadas que puede utilizar en una situación habitual de Amazon DocumentDB, consulte Cree una IPv4-only VPC para usarla con un clúster de DocumentDB.

Direcciones IPv6

De manera opcional, puede asociar un bloque de CIDR IPv6 a su VPC y sus subredes y asignar direcciones IPv6 desde dicho bloque a los recursos de su VPC. Cada dirección IPv6 es única a nivel mundial.

El bloque de CIDR IPv6 de su VPC se asigna automáticamente de entre el grupo de direcciones IPv6 de Amazon. Usted no puede elegir el rango.

Al conectarse a una dirección IPv6, asegúrese de que se cumplan las siguientes condiciones:

- El cliente se ha configurado de manera que se permita el tráfico de la base de datos a través de IPv6.
- Los grupos de seguridad de DocumentDB utilizados por el clúster están configurados correctamente para permitir el tráfico del cliente a la base de datos a través de IPv6.
- La pila del sistema operativo del cliente permite el tráfico en la dirección IPv6. Además, los controladores y bibliotecas del sistema operativo están configurados para elegir el punto de conexión del clúster predeterminado correcto (IPv4 o IPv6).

Para obtener más información sobre IPv6, consulte Direccionamiento IP en la Guía del usuario de Amazon Virtual Private Cloud.

Dual-stack modo

Cuando un clúster puede comunicarse a través de los protocolos de direcciones tanto IPv4 como IPv6, se ejecuta en modo de pila doble. Por lo tanto, los recursos pueden comunicarse con el clúster a través de IPv4, IPv6 o ambos. DocumentDB deshabilita el acceso a la puerta de enlace de Internet para los puntos de conexión IPv6 de clústeres privados en modo de pila doble. DocumentDB hace esto para garantizar que los puntos de conexión IPv6 sean privados y solo se pueda acceder a ellos desde la VPC.

Temas

- Dual-stack modo y grupos de subredes
- Trabajo con clústeres en modo de pila doble
- Modificar IPv4-only los clústeres para usar el modo de doble pila
- Dual-stack modo: disponibilidad de regiones y versiones.
- Limitaciones de los clústeres de red de pila doble

Dual-stack modo y grupos de subredes

Para utilizar el modo de pila doble, asegúrese de que cada subred del grupo de subredes que asocie al clúster tenga un bloque de CIDR de IPv6 asociado. Puede crear un nuevo grupo de subredes o modificar un grupo existente de subredes para cumplir este requisito. Cuando un clúster esté en modo de pila doble, los clientes podrán conectarse como siempre. Asegúrese de que los firewalls de seguridad del cliente y los grupos de seguridad del clúster de DocumentDB estén configurados correctamente para permitir el tráfico a través de IPv6. Para conectarse, los clientes utilizan el punto de conexión del clúster. Las aplicaciones de cliente pueden especificar qué protocolo prefieren al conectarse a una base de datos. En modo de pila doble, el clúster detecta el protocolo de red preferido del cliente (IPv4 o IPv6) y utiliza ese protocolo para la conexión.

Si un grupo de subredes deja de admitir el modo de pila doble debido a la eliminación de subredes o a la disociación de CIDR, existe el riesgo de que se produzca un estado de red incompatible para los clústeres asociados al grupo de subredes. Además, no puede utilizar el grupo de subredes cuando crea un clúster nuevo en modo de pila doble.

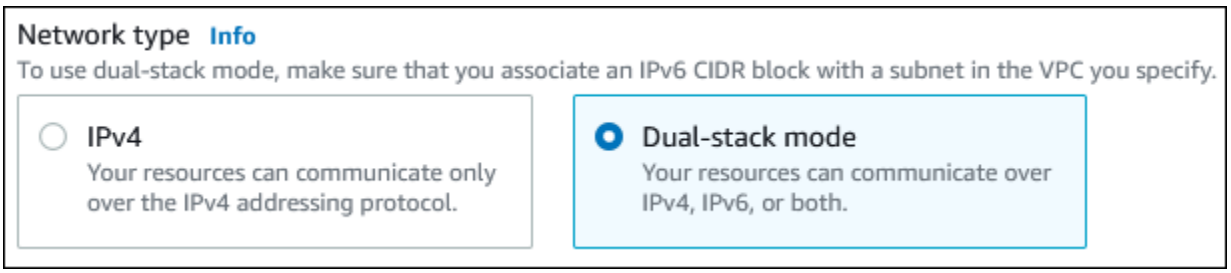
Para determinar si un grupo de subredes admite el modo de doble pila Consola de administración de AWS, consulte el tipo de red en la página de detalles del grupo de subredes. Para determinar si un grupo de subredes admite el modo de doble pila AWS CLI, ejecute el `describe-db-subnet-groups` comando y vea el resultado. `SupportedNetworkTypes`

Las réplicas de lectura se tratan como clústeres independientes y pueden tener un tipo de red diferente al del clúster principal. Si cambia el tipo de red del clúster principal de una réplica de lectura, la réplica de lectura no se verá afectada. Cuando restaura un clúster, puede restaurarla a cualquier tipo de red compatible.

Trabajo con clústeres en modo de pila doble

Cuando crea o modifica un clúster, puede especificar que el modo de pila doble permita que los recursos se comuniquen con el clúster a través de IPv4, IPv6 o ambos.

Cuando lo usas Consola de administración de AWS para crear o modificar un clúster, puedes especificar el modo de doble pila en la sección Tipo de red. En la imagen siguiente se muestra la sección Tipo de red en la consola:



Cuando utilice el AWS CLI para crear o modificar un clúster, defina la `--network-type` opción `DUAL` para usar el modo de doble pila. Si utiliza la API de DocumentDB para crear o modificar un clúster, establezca el parámetro `NetworkType` en `DUAL` para utilizar el modo de pila doble. Si el modo de pila doble no es compatible con la versión del motor ni el grupo de subredes de DocumentDB especificados, se devuelve el error `NetworkTypeNotSupported`.

Para obtener más información acerca de la creación de un clúster, consulte Creación de un clúster de Amazon DocumentDB. Para obtener más información sobre la modificación de un clúster, consulteModificación de un clúster de Amazon DocumentDB.

Para determinar si un clúster está en modo de pila doble mediante la consola, consulte Tipo de red en la pestaña Conectividad y seguridad del clúster.

Modificar IPv4-only los clústeres para usar el modo de doble pila

Puede modificar un IPv4-only clúster para que utilice el modo de doble pila. Para ello, cambie el tipo de red del clúster.

Se recomienda cambiar el tipo de red del clúster de Amazon DocumentDB durante un período de mantenimiento. Puede configurar el tipo de red manualmente mediante el comando `modify-db-cluster`.

Antes de modificar un clúster para utilizar el modo de pila doble, asegúrese de que su grupo de subredes admite el modo de pila doble. Si el grupo de subredes asociado al clúster no admite el modo de pila doble, especifique otro grupo de subredes que lo admita cuando modifique el clúster. La modificación del grupo de subredes de un clúster puede provocar un tiempo de inactividad.

Si modifica el grupo de subredes de un clúster antes de cambiar el clúster para utilizar el modo de pila doble, asegúrese de que el grupo de subredes sea válido para el clúster antes del cambio y después de este.

Ejecute la `ModifyDBCluster` llamada solo con el `NetworkType` parámetro establecido `DUAL` para cambiar la red al modo de doble pila. Si se añaden otros parámetros junto con `NetworkType`

en la misma llamada a la API, se podría producir un tiempo de inactividad. Para modificar varios parámetros, asegúrese de que la modificación del tipo de red se haya completado correctamente antes de enviar otra solicitud de `ModifyDBCluster` con otros parámetros.

Si no puede conectarse al clúster después del cambio, asegúrese de que los firewalls de seguridad de la base de datos y del cliente y las tablas de enrutamiento se hayan configurado correctamente para permitir el tráfico a la base de datos de la red seleccionada (IPv4 o IPv6). Es posible que también tenga que modificar los parámetros, las bibliotecas o los controladores del sistema operativo para conectarse mediante una dirección IPv6.

Para modificar un IPv4-only clúster para que utilice el modo de doble pila

1. Modifique un grupo de subredes para admitir el modo de pila doble o cree un grupo de subredes que admita el modo de pila doble:

   a. Asocie un bloque de CIDR IPv6 a su VPC.

      Para obtener las instrucciones, consulte Adición o eliminación de un bloque de CIDR de su VPC en la Guía del usuario de Amazon VPC.

   b. Adjunte el bloque de CIDR IPv6 a todas las subredes de su grupo de subredes.

      Para obtener las instrucciones, consulte Adición o eliminación de un bloque de CIDR de IPv6 en su subred en la Guía del usuario de Amazon Virtual Private Cloud.

   c. Confirme que el grupo de subredes admita el modo de pila doble.

      Si utiliza el Consola de administración de AWS, seleccione el grupo de subredes y asegúrese de que el valor de los tipos de red compatibles sea Doble.

      Si utiliza el AWS CLI, ejecute el `describe-db-subnet-groups` comando y asegúrese de que el `SupportedNetworkType` valor del clúster sea`Dual`.

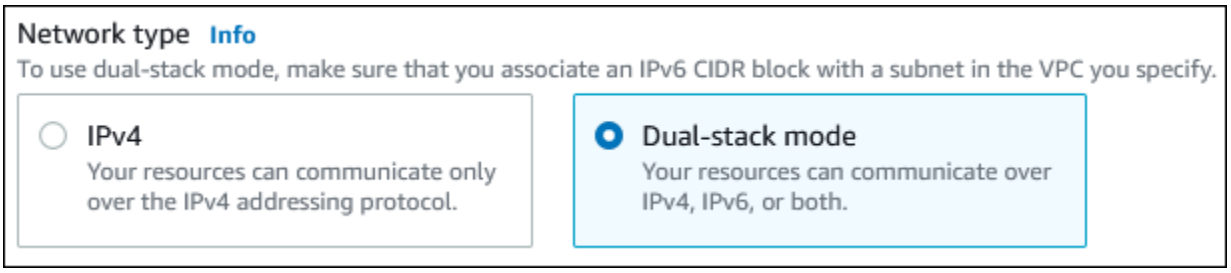
2. Modifique el grupo de seguridad asociado al clúster para permitir conexiones IPv6 a la base de datos o cree un nuevo grupo de seguridad que permita conexiones IPv6.

   Para obtener más información, consulte Reglas del grupo de seguridad en la Guía del usuario de Amazon Virtual Private Cloud.

3. Modifique el clúster para admitir el modo de pila doble. Para ello, defina el tipo de red en Dual-stack modo.

   Si utiliza la consola, asegúrese de que la siguiente configuración sea correcta:

- Tipo de red: Dual-stack modo



- Grupo de subredes: el grupo de subredes que configuró en un paso anterior.
- Grupo de seguridad: la seguridad que configuró en el paso anterior.

Si está utilizando el AWS CLI, asegúrese de que los siguientes ajustes son correctos:

- `--network-type` — `dual`
- `--db-subnet-group-name`: el grupo de subredes que configuró en un paso anterior.
- `--vpc-security-group-ids`: el grupo de seguridad de la VPC que configuró en un paso anterior.

Por ejemplo:

```
aws docdb modify-db-cluster --db-cluster-identifier <cluster-name> --network-type
 "DUAL"
```

4. Confirme que el clúster admite el modo de pila doble.

   Si utiliza la consola, elija la pestaña Conectividad y seguridad del clúster y asegúrese de que el valor del tipo de red sea Dual-stack modo.

   Si está utilizando el AWS CLI, ejecute el `describe-db-cluster` comando y asegúrese de que el `NetworkType` valor del clúster sea`dual`.

   Ejecute el comando `dig` en el punto de conexión del clúster para identificar la dirección IPv6 que tiene asociada.

   ```
   dig <db-cluster-endpoint> AAAA
   ```

   Utilice el punto de conexión del clúster (no la dirección IPv6) para conectarse al clúster.

Dual-stack modo: disponibilidad de regiones y versiones.

La disponibilidad y la compatibilidad de las características varían en las Regiones de AWS.

Compatibilidad de regiones

El modo de doble pila para Amazon DocumentDB está disponible actualmente en todas las versiones comerciales Regiones de AWS.

Soporte de versiones

Dual-stack el modo es compatible con las versiones 4.0, 5.0 y 8.0 de Amazon DocumentDB. Si no puede acceder al modo de doble pila en ninguna de estas versiones, asegúrese de que está ejecutando la última versión del parche del motor en su clúster.

Limitaciones de los clústeres de red de pila doble

Las siguientes limitaciones se aplican a los clústeres de red de pila doble:

- Los clústeres no pueden utilizar el protocolo IPv6 exclusivamente. Pueden utilizar IPv4 exclusivamente o utilizar el protocolo IPv4 y IPv6 (modo de pila doble).
- Amazon DocumentDB no admite subredes IPv6 nativas.
- Los clústeres que utilizan el modo de pila doble deben ser de tipo privado. No pueden ser accesibles públicamente.

Creación de un clúster en una VPC

Los siguientes procedimientos le permiten crear un clúster en una VPC. Para utilizar la VPC predeterminada, puede comenzar con el paso 2 y utilizar la VPC y el grupo de subredes que ya se han creado para usted. Además, también puede crear VPC adicionales, de ser necesario.

> **Note**
>
> Si desea que el clúster de la VPC sea accesible públicamente, debe actualizar la información de DNS para la VPC. Para ello, active los atributos `DNS hostnames` y `DNS resolution` de la VPC. Para obtener más información sobre la actualización del DNS de una instancia de VPC, consulte Ver y actualizar los atributos de DNS de su VPC en la Guía del usuario de Amazon Virtual Private Cloud.

Para crear un clúster en una VPC, siga estos pasos:

- Paso 1: Crear una VPC
- Paso 2: crear un grupo de subredes
- Paso 3: Crear un grupo de seguridad de VPC
- Paso 4: crear un clúster en la VPC

Paso 1: Crear una VPC

Cree una VPC con subredes en al menos dos zonas de disponibilidad. Utilizará estas subredes cuando cree un grupo de subredes. Si tiene una VPC predeterminada, se crea automáticamente una subred en cada zona de disponibilidad de la Región de AWS.

Para obtener más información, consulte Cree una IPv4-only VPC para usarla con un clúster de DocumentDB o consulte Creación de una VPC en la Guía del usuario de Amazon Virtual Private Cloud.

Paso 2: crear un grupo de subredes

Los grupos de subredes son una colección de subredes (normalmente privadas) que crea para una VPC y que después asigna a los clústeres. Un grupo de subredes te permite especificar una VPC concreta al crear clústeres mediante la API AWS CLI o DocumentDB. Si usa la Consola de administración de AWS, solo tiene que elegir la VPC y las subredes que desea usar. Cada grupo de subredes debe tener como mínimo una subred en, al menos, dos zonas de disponibilidad de la Región de AWS. Como práctica recomendada, cada grupo de subredes debería tener al menos una subred por cada una de las zonas de disponibilidad en la Región de AWS.

Para que un clúster sea accesible públicamente, las subredes del grupo de subredes deben tener una puerta de enlace de Internet. Para obtener más información sobre las puertas de enlace de Internet para las subredes, consulte Activación del acceso a Internet de una VPC con una puerta de enlace de Internet en la Guía del usuario de Amazon Virtual Private Cloud.

> **Note**
>
> El grupo de subredes para una zona local puede tener solo una subred.

Cuando crea un clúster en una VPC, puede elegir un grupo de subredes. Amazon DocumentDB elige una subred y una dirección IP (dentro de esa subred) para asociarla con el clúster. Si no existen

grupos de subredes, Amazon DocumentDB crea un grupo de subredes predeterminado cuando crea el clúster. DocumentDB crea y asocia una interfaz de red elástica a su clúster con esa dirección IP. El clúster utiliza la zona de disponibilidad que contiene la subred.

En este paso, debe crear un grupo de subredes y debe agregar las subredes que creó para la VPC.

Para crear un grupo de subredes

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, elija Subnet groups (grupos de subredes).
3. Seleccione Crear.
4. En Nombre, escriba el nombre del grupo de subredes.
5. En Descripción, escriba la descripción del grupo de subredes.
6. En la sección Añadir subredes, para VPC, elija la VPC predeterminada o la VPC que creó. A continuación, elija las zonas de disponibilidad que incluyen las subredes en Zonas de disponibilidad, y, luego, elija las subredes en Subredes.
7. Seleccione Crear.

   El nuevo grupo de subredes aparece en la lista de Grupos de subredes de la consola de DocumentDB. Puede elegir el grupo de subredes para ver los detalles, incluidas todas las subredes asociadas al grupo, en el panel de detalles de la parte inferior de la ventana.

Paso 3: Crear un grupo de seguridad de VPC

Antes de crear el clúster, cree un grupo de seguridad de la VPC para asociarlo a él. Si no crea un grupo de seguridad de VPC, puede utilizar el grupo de seguridad predeterminado cuando cree el clúster. Para obtener instrucciones sobre cómo crear un grupo de seguridad para su clústerCree una IPv4-only VPC para usarla con un clúster de DocumentDB, consulte o consulte Controle el tráfico hacia sus AWS recursos mediante grupos de seguridad en la Guía del usuario de Amazon Virtual Private Cloud.

Paso 4: crear un clúster en la VPC

En este paso, se crea un clúster con el nombre de la VPC, el grupo de subredes y el grupo de seguridad de la VPC que creó en los pasos anteriores.

> **Note**
>
> Si desea que un clúster de la VPC sea accesible públicamente, debe activar los atributos `DNS hostnames` y `DNS resolution` de la VPC. Para obtener más información, consulte Ver y actualizar los atributos de DNS de su VPC en la Guía del usuario de Amazon Virtual Private Cloud.

Para obtener información detallada acerca de cómo crear un clúster, consulte Creación de un clúster de Amazon DocumentDB.

Cuando la sección Conectividad se lo pida, introduzca el nombre de la VPC, el grupo de subredes y el grupo de seguridad de la VPC.

> **Note**
>
> La actualización de VPC no se admite actualmente para los clústeres de DocumentDB.

Acceso a un clúster de Amazon DocumentDB en una VPC

Amazon DocumentDB admite las siguientes situaciones para obtener acceso a un clúster en una VPC:

Temas

- Acceso a un clúster en una VPC desde una instancia de Amazon EC2 de la misma VPC
- Acceso a un clúster de una VPC desde una instancia de Amazon EC2 de otra VPC

Acceso a un clúster en una VPC desde una instancia de Amazon EC2 de la misma VPC

Un uso común de un clúster en una VPC es compartir datos con un servidor de aplicaciones que se ejecuta en una instancia de Amazon EC2 de la misma VPC.

La forma más sencilla de administrar el acceso entre instancias EC2 y clústeres de la misma VPC es realizar lo siguiente:

- Cree un grupo de seguridad de VPC para sus clústeres. Este grupo de seguridad se puede utilizar para restringir el acceso a los clústeres. Por ejemplo, puede crear una regla personalizada para este grupo de seguridad. Esto puede permitir el acceso TCP mediante el uso del puerto que asignó al clúster cuando lo creó y una dirección IP que utiliza para acceder al clúster para el desarrollo u otras finalidades.
- Cree el grupo de seguridad de VPC al que pertenecerán las instancias EC2 (clientes y servidores web). Este grupo de seguridad puede, si es necesario, permitir el acceso a la instancia EC2 desde Internet a través de la tabla de enrutamiento de la VPC. Por ejemplo, puede establecer reglas en este grupo de seguridad para permitir el acceso mediante TCP a la instancia EC2 a través del puerto 22.
- Cree reglas personalizadas en el grupo de seguridad para sus clústeres, que permitan las conexiones desde el grupo de seguridad que creó para las instancias EC2. Estas reglas podrían permitir que cualquier miembro del grupo de seguridad acceda a los clústeres.

Hay una subred pública y privada adicional en una zona de disponibilidad independiente. Los grupos de subredes de DocumentDB requieren una subred en, al menos, dos zonas de disponibilidad. La subred adicional facilita el cambio a una implementación de clúster multi-AZ en el futuro.

Para conocer las instrucciones sobre cómo crear una VPC con subredes pública y privada para esta situación, consulte Cree una IPv4-only VPC para usarla con un clúster de DocumentDB.

> **Tip**
>
> Puede configurar la conectividad de red entre una instancia de Amazon EC2 y un clúster de DocumentDB automáticamente cuando crea el clúster. Para obtener más información, consulte Conexión automática de Amazon EC2.

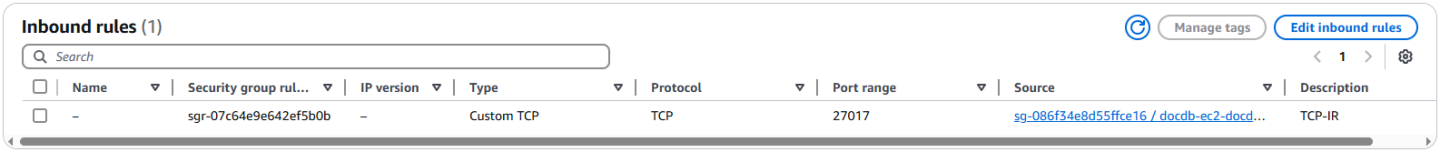
Para crear una regla en un grupo de seguridad de VPC que permita las conexiones desde otro grupo de seguridad, haga lo siguiente:

1. Inicie sesión en la consola de Amazon VPC Consola de administración de AWS y ábrala en. https://console.aws.amazon.com/vpc
2. En el panel de navegación, ubique y elija Grupos de seguridad.
3. Elija o cree el grupo de seguridad al que desea que puedan tener acceso los miembros de otro grupo de seguridad. Este es el grupo de seguridad que usa para los clústeres. Elija la pestaña

Inbound Rules (Reglas de entrada) y, a continuación, elija Edit inbound rules (Editar reglas de entrada).

4. En la página Edit inbound rules (Editar reglas de entrada), elija Add Rule (Agregar regla).
5. En Tipo, elija la entrada que corresponda al puerto que utilizó cuando creó el clúster, como TCP personalizada.
6. En el campo Origen, comience a escribir el identificador del grupo de seguridad, que enumera los grupos de seguridad coincidentes. Elija el grupo de seguridad cuyos miembros desea que tengan acceso a los recursos protegidos por este grupo de seguridad. En el escenario anterior, este es el grupo de seguridad que utiliza para su instancia EC2.
7. Si es necesario, repita los pasos para el protocolo TCP y cree una regla con Todo TCP como Tipo y el grupo de seguridad en el campo Origen. Si va a utilizar el protocolo UDP, cree una regla con All UDP (Todo UDP) en el campo Type (Tipo) y con el grupo de seguridad en el campo Source (Origen).
8. Seleccione Guardar reglas.

La siguiente pantalla muestra una regla de entrada con un grupo de seguridad para su origen.



Para obtener más información sobre cómo conectarse al clúster desde su instancia de EC2, consulte Conexión automática de Amazon EC2.

Acceso a un clúster de una VPC desde una instancia de Amazon EC2 de otra VPC

Cuando los clústeres están en una VPC que no coincide con la de la instancia de EC2 que se está utilizando para obtener acceso a ella, puede usar la interconexión con VPC para obtener acceso al clúster.

Una conexión de emparejamiento de VPC es una conexión de redes entre dos VPC que permite direccionar el tráfico entre ellas mediante direcciones IP privadas. Los recursos de ambas VPC se pueden comunicar entre sí siempre que se encuentren en la misma red. Puede crear una conexión de interconexión de VPC entre sus propias VPC, con una VPC en otra AWS cuenta o con una VPC en otra. Región de AWS Para obtener más información sobre las interconexiones de VPC, consulte Interconexión con VPC en la Guía de usuario de Amazon Virtual Private Cloud.

Cree una IPv4-only VPC para usarla con un clúster de DocumentDB

Una situación común incluye un clúster en una nube privada virtual (VPC) basada en el servicio Amazon VPC. Por ejemplo, esta VPC podría compartir datos con un servicio o aplicación que se ejecute en la misma VPC. En este tema, creará la VPC para esta situación.

Temas

- Paso 1: crear una VPC con subredes públicas y privadas
- Paso 2: crear un grupo de seguridad de VPC para una aplicación pública
- Paso 3: crear un grupo de seguridad de VPC para un clúster privado
- Paso 4: Crear un grupo de subredes
- Eliminación de una VPC

Su clúster debe estar disponible únicamente para su servidor web, y no para la Internet pública. Además, cree de una VPC con subredes públicas y privadas. La aplicación se aloja en la subred pública, para que pueda obtener acceso a la Internet pública. El clúster se aloja en una subred privada. La aplicación puede conectarse al clúster porque se aloja en la misma VPC. Sin embargo, el clúster no está disponible en la red pública de internet, lo que proporciona mayor seguridad.

Con el procedimiento de este tema se configura una subred pública y privada adicional en una zona de disponibilidad independiente. El procedimiento no utiliza estas subredes. Los grupos de subredes de DocumentDB requieren una subred en, al menos, dos zonas de disponibilidad. La subred adicional facilita la configuración de más de una instancia de DocumentDB.

En este tema se describe la configuración de una VPC para clústeres de Amazon DocumentDB. Para obtener más información sobre Amazon VPC, consulte la Guía del usuario de Amazon VPC.

> **Tip**
>
> Puede configurar la conectividad de red entre una instancia de Amazon EC2 y un clúster de DocumentDB automáticamente cuando crea el clúster. La configuración de red es similar a la que se describe en esta situación. Para obtener más información, consulte Conexión automática de Amazon EC2.

Paso 1: crear una VPC con subredes públicas y privadas

Utilice el siguiente procedimiento para crear una VPC con subredes públicas y privadas.

Para crear una VPC y subredes

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.
2. En la esquina superior derecha del Consola de administración de AWS, elige la región en la que vas a crear tu VPC. En este ejemplo se utiliza la región Oeste de EE. UU. (Oregón).
3. En la esquina superior izquierda, elija VPC Dashboard (Panel de control VPC). Para comenzar a crear una VPC, elija Create VPC (Crear una VPC).
4. En Resources to create (Recursos para crear), en VPC settings (Configuración VPC), elija VPC and more (VPC y más).
5. En VPC settings (Configuración de la VPC), establezca estos valores:
   - Generación automática de etiquetas identificativas: **`example`**
   - Bloque CIDR de IPv4 — **`10.0.0.0/16`**
   - Bloque de CIDR IPv6: ningún bloque de CIDR IPv6
   - Tenencia: predeterminada
   - Cantidad de zonas de disponibilidad (AZ): 2
   - Personalizar AZ: conserve los valores predeterminados.
   - Cantidad de subredes públicas: 2
   - Cantidad de subredes privadas: 2
   - Personalizar bloques de CIDR de subredes: conserve los valores predeterminados.
   - Puertas de enlace NAT ($): ninguna
   - Puntos de conexión de VPC: ninguno
   - Opciones de DNS: conserve los valores predeterminados.
6. Seleccione Creación de VPC.

Paso 2: crear un grupo de seguridad de VPC para una aplicación pública

A continuación, cree un grupo de seguridad para el acceso público. Para conectarse a instancias de EC2 públicas en su VPC, añada reglas de entrada a su grupo de seguridad de VPC. Permiten que el tráfico se conecte desde Internet.

Para crear un grupo de seguridad de VPC

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.
2. Elija VPC Dashboard (Panel VPC), seguido de Security Groups (Grupos de seguridad) y, por último, Create Security Group (Crear grupo de seguridad).
3. En la página Create Security Group (Crear grupo de seguridad), establezca estos valores:
   - Nombre del grupo de seguridad: **`example-securitygroup`**
   - Descripción — **`Application security group`**
   - VPC: elija la VPC que creó anteriormente, por ejemplo: vpc-example.
4. Agregar reglas de entrada al grupo de seguridad
   a. Determine la dirección IP que usará para conectarse a las instancias de EC2 mediante Secure Shell (SSH). Para determinar su dirección IP pública, en otra ventana o pestaña del navegador, puede utilizar el servicio en https://checkip.amazonaws.com. Un ejemplo de dirección IP es `203.0.113.25/32`.

      En muchos casos, puede conectarse a través de un proveedor de servicios de internet (ISP) o protegido por un firewall sin una dirección IP estática. Si es así, busque el rango de direcciones IP utilizadas por los ordenadores cliente.

      > **Warning**
      >
      > Si utiliza `0.0.0.0/0` para el acceso SSH, permita que todas las direcciones IP accedan a sus instancias públicas mediante SSH. Este método es aceptable para un periodo de tiempo corto en un entorno de prueba, pero no es seguro en entornos de producción. En entornos de producción, solo debe autorizar una dirección IP específica o un intervalo de direcciones para acceder a sus instancias mediante SSH.

   b. En la sección Inbound rules (Reglas de entrada), elija Add rule (agregar regla).
   c. Establezca los siguientes valores para la regla de entrada nueva con objeto de permitir el acceso SSH a la instancia de Amazon EC2. Si lo hace, podrá conectarse a la instancia de EC2 para instalar aplicaciones y otras utilidades. También puede conectarse a la instancia de EC2 para cargar contenido para su aplicación.
      - Escriba — **`SSH`**

- Origen: la dirección IP o el rango de direcciones que creó en el Paso a, por ejemplo: **`203.0.113.25/32`**.

d. Seleccione Agregar regla.

e. Establezca los siguientes valores para la regla de entrada nueva con objeto de permitir el acceso HTTP a su aplicación.

- Tipo — **`HTTP`**
- Fuente — **`0.0.0.0/0`**

5. Para crear el grupo de seguridad, elija Create security group (Crear grupo de seguridad).

   Anote el identificador del grupo de seguridad, ya que lo necesitará más tarde en otro procedimiento.

Paso 3: crear un grupo de seguridad de VPC para un clúster privado

Para crear un clúster privado, debe crear un segundo grupo de seguridad para el acceso privado. Para conectarse a clústeres privados en la VPC, agregue reglas de entrada al grupo de seguridad de la VPC que permitan el tráfico solo desde su aplicación.

Para crear un grupo de seguridad de VPC

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.
2. Elija VPC Dashboard (Panel VPC), seguido de Security Groups (Grupos de seguridad) y, por último, Create Security Group (Crear grupo de seguridad).
3. En la página Create Security Group (Crear grupo de seguridad), establezca estos valores:

   - Nombre del grupo de seguridad: **`example-securitygroup`**
   - Descripción — **`Instance security group`**
   - VPC: elija la VPC que creó anteriormente, por ejemplo: vpc-example.

4. Agregar reglas de entrada al grupo de seguridad

   a. En la sección Inbound rules (Reglas de entrada), elija Add rule (gregar regla).

   b. Establezca los siguientes valores para la regla de entrada nueva a fin de permitir el tráfico de DocumentDB en el puerto 27017 desde la instancia de Amazon EC2. Si lo hace, podrá conectarse desde su aplicación a su clúster. Si lo hace, puede almacenar y recuperar datos desde su aplicación a su base de datos.

- Tipo — **`Custom TCP`**
- Origen: por ejemplo, el identificador del grupo de seguridad de la aplicación que creó anteriormente en este tema: sg-9edd5cfb.

c. Seleccione Agregar regla.

d. Establezca los siguientes valores para la regla de entrada nueva con objeto de permitir el acceso HTTP a su aplicación.

- Tipo — **`HTTP`**
- Fuente — **`0.0.0.0/0`**

5. Para crear el grupo de seguridad, elija Create security group (Crear grupo de seguridad).

Paso 4: Crear un grupo de subredes

Un grupo de subredes es una colección de subredes que se crean en una VPC y que después se asignan a los clústeres. Los grupos de subredes le permiten especificar una VPC en particular cuando crea los clústeres.

Para crear un grupo de subredes

1. Identifique las subredes privadas de la base de datos en la VPC.

   a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.

   b. Seleccione VPC Dashboard (Panel de control de VPC) y, a continuación, seleccione Subnets (Subredes).

   c. Anote los identificadores de subred de las subredes que creó en el Paso 1 denominadas, por ejemplo: example-subnet-private1-us-west-2a y example-subnet-private2-us-west-2b. Necesitará los identificadores de subred cuando cree el grupo de subredes.

2. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb

   Asegúrese de conectarse a la consola de Amazon DocumentDB, no a la consola de Amazon VPC.

3. En el panel de navegación, elija Subnet groups (grupos de subredes).

4. Seleccione Crear.

5. En la página Crear grupo de subredes, establezca estos valores en la sección Detalles del grupo de subredes:
   - Nombre — **`example-db-subnet-group`**
   - Descripción — **`Instance security group`**
6. En la sección Configuración, establezca los siguientes valores:
   - VPC: elija la VPC que creó anteriormente, por ejemplo: vpc-example.
   - Zonas de disponibilidad: seleccione las dos zonas de disponibilidad creadas en el Paso 1. Ejemplo: us-west-2a y us-west-2b.
   - Subredes: elija las subredes privadas que creó en el Paso 1.
7. Seleccione Crear.

El nuevo grupo de subredes aparece en la lista de grupos de subredes de la consola de DocumentDB. Puede elegir el grupo de subredes para ver los detalles en el panel de detalles. Estos detalles incluyen todas las subredes asociadas al grupo.

> **Note**
>
> Si creó esta VPC para asociarla a un clúster de DocumentDB, cree el clúster siguiendo las instrucciones que se indican en Creación de un clúster de Amazon DocumentDB.

Eliminación de una VPC

Puede eliminar una VPC y los demás recursos que se utilizan en ella si ya no son necesarios.

> **Note**
>
> Si agregó recursos en la VPC que creó en este tema, es posible que primero tenga que eliminar estos para poder eliminar la VPC. Por ejemplo, estos recursos pueden incluir instancias de Amazon EC2 o clústeres de DocumentDB. Para obtener más información, consulte Eliminación de la VPC en la Guía del usuario de Amazon VPC.

Para eliminar una VPC y los recursos relacionados

1. Elimine el grupo de subredes:

a. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb

b. En el panel de navegación, elija Subnet groups (grupos de subredes).

c. Seleccione el grupo de subredes que desea eliminar, como example-db-subnet-group.

d. Elija Delete (Eliminar) y, a continuación, elija Delete (Eliminar) en la ventana de confirmación.

2. Anote el ID de la VPC:

a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.

b. Seleccione Panel de VPC y, a continuación, seleccione Sus VPC.

c. En la lista, identifique la VPC que creó, por ejemplo, vpc-example.

d. Anote el ID de la VPC que ha creado. Necesitará el ID de la VPC en pasos posteriores.

3. Elimine los grupos de seguridad:

a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.

b. Seleccione VPC Dashboard (Panel de control de VPC) y, a continuación, seleccione Security Groups (Grupos de seguridad).

c. Seleccione el grupo de seguridad del clúster de Amazon DocumentDB, por ejemplo, example-securitygroup.

d. En Acciones, elija Eliminar grupos de seguridad y, a continuación, seleccione Eliminar en el diálogo de confirmación.

e. En la página Grupos de seguridad, seleccione el grupo de seguridad que corresponda a la instancia de Amazon EC2, por ejemplo, example-securitygroup.

f. En Acciones, elija Eliminar grupos de seguridad y, a continuación, seleccione Eliminar en el diálogo de confirmación.

4. Elimine la VPC:

a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.

b. Seleccione Panel de VPC y, a continuación, seleccione Sus VPC.

c. Seleccione la VPC que desea eliminar, por ejemplo, vpc-example.

d. En Acciones, elija Eliminar VPC.

La página de confirmación muestra otros recursos asociados a la VPC que también se eliminarán, incluidas las subredes asociadas a ella.

e. En el diálogo de confirmación, introduzca **`delete`** y, a continuación, elija Eliminar.

Creación de una VPC de pila doble para utilizarla con un clúster de DocumentDB

Una situación común incluye un clúster en una nube privada virtual (VPC) basada en el servicio Amazon VPC. Esta VPC comparte datos con una instancia de Amazon EC2 pública que se ejecuta en la misma VPC. En este tema, creará la VPC para esta situación.

Temas

- Paso 1: crear una VPC con subredes públicas y privadas
- Paso 2: crear un grupo de seguridad de la VPC para una instancia de Amazon EC2 pública
- Paso 3: crear un grupo de seguridad de VPC para un clúster privado
- Paso 4: Crear un grupo de subredes
- Paso 5: crear una instancia de Amazon EC2 en el modo de pila doble
- Paso 6: crear un clúster en modo de pila doble
- Paso 7: conectarse a la instancia de Amazon EC2 y al clúster de base de datos
- Elimine la VPC

En este procedimiento, se crea la VPC para este escenario que funciona con una base de datos que se ejecuta en modo de doble pila. Dual-stack el modo permite la conexión a través del protocolo de direccionamiento IPv6. Para obtener más información sobre el direccionamiento de IP, consulte Direcciones IP de Amazon DocumentDB.

Dual-stack los clústeres de red son compatibles en la mayoría de las regiones. Para obtener más información, consulte Dual-stack modo: disponibilidad de regiones y versiones.. Para ver las limitaciones del modo de doble pila, consulte Limitaciones de los clústeres de red de pila doble.

Este tema y el IPv4-only tema crean las subredes públicas y privadas en la misma VPC. Para obtener información sobre cómo crear el clúster de Amazon DocumentDB en una VPC y la instancia de Amazon EC2 en una VPC diferente, consulte Acceso a un clúster de Amazon DocumentDB en una VPC.

Su clúster debe estar disponible solamente para su instancia de Amazon EC2, y no para la Internet pública. Además, cree de una VPC con subredes públicas y privadas. La instancia de Amazon EC2

está alojada en la subred pública, para que pueda acceder a la Internet pública. El clúster se aloja en una subred privada. La instancia de EC2 puede conectarse al clúster porque se aloja en la misma VPC. Sin embargo, el clúster no está disponible en la Internet pública, lo que proporciona mayor seguridad.

Con los procedimientos de este tema, se configura una subred pública y privada adicional en una zona de disponibilidad independiente. El procedimiento no utiliza estas subredes. Los grupos de subredes de DocumentDB requieren una subred en, al menos, dos zonas de disponibilidad. La subred adicional facilita la configuración de más de una instancia de DocumentDB.

Para crear un clúster que utilice el modo de doble pila, especifique el Dual-stack modo e en la configuración de tipo de red. También puede modificar un clúster con el mismo ajuste. Para obtener más información acerca de la creación de un clúster, consulte Creación de un clúster de Amazon DocumentDB. Para obtener más información sobre la modificación de un clúster de base de datos, consulte Modificación de un clúster de Amazon DocumentDB.

En este tema se describe la configuración de una VPC para clústeres de Amazon DocumentDB. Para obtener más información sobre Amazon VPC, consulte la Guía del usuario de Amazon VPC.

Paso 1: crear una VPC con subredes públicas y privadas

Utilice el siguiente procedimiento para crear una VPC con subredes públicas y privadas.

Para crear una VPC y subredes

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.
2. En la esquina superior derecha del Consola de administración de AWS, elige la región en la que vas a crear tu VPC. En este ejemplo se utiliza la región Oeste de EE. UU. (Oregón).
3. En la esquina superior izquierda, elija VPC Dashboard (Panel de control VPC). Para comenzar a crear una VPC, elija Create VPC (Crear una VPC).
4. En Resources to create (Recursos para crear), en VPC settings (Configuración VPC), elija VPC and more (VPC y más).
5. En VPC settings (Configuración de la VPC), establezca estos valores:
   - Generación automática de etiquetas identificativas: **`example-dual-stack`**
   - Bloque CIDR de IPv4 — **`10.0.0.0/16`**
   - Bloque CIDR IPv6 — bloque CIDR IPv6 Amazon-provided

- Tenencia: predeterminada
- Cantidad de zonas de disponibilidad (AZ): 2
- Personalizar AZ: conserve los valores predeterminados.
- Cantidad de subredes públicas: 2
- Cantidad de subredes privadas: 2
- Personalizar bloques de CIDR de subredes: conserve los valores predeterminados.
- Puertas de enlace NAT ($): ninguna
- Puerta de enlace de Internet solo de salida: no
- Puntos de conexión de VPC: ninguno
- Opciones de DNS: conserve los valores predeterminados.

6. Seleccione Creación de VPC.

Paso 2: crear un grupo de seguridad de la VPC para una instancia de Amazon EC2 pública

Primero debe crear un grupo de seguridad para el acceso público. Para conectarse a instancias EC2 públicas, añada reglas de entrada al grupo de seguridad de la VPC que permitan el tráfico para las conexiones desde internet.

Para crear un grupo de seguridad de VPC

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.
2. Elija VPC Dashboard (Panel VPC), seguido de Security Groups (Grupos de seguridad) y, por último, Create Security Group (Crear grupo de seguridad).
3. En la página Create Security Group (Crear grupo de seguridad), establezca estos valores:
   - Nombre del grupo de seguridad: **`example-dual-stack-securitygroup`**
   - Descripción — **`Dual-stack security group`**
   - VPC: elija la VPC que creó anteriormente, por ejemplo: vpc-example-dual-stack.
4. Agregar reglas de entrada al grupo de seguridad
   a. Determine la dirección IP que usará para conectarse a las instancias de EC2 mediante Secure Shell (SSH). Para determinar su dirección IP pública, en otra ventana o pestaña del navegador, puede utilizar el servicio en https://checkip.amazonaws.com.

Un ejemplo de un rango de direcciones del Protocolo de Internet versión 4 (IPv4) es`203.0.113.25/32`. Un ejemplo de rango de direcciones del protocolo de internet versión 6 (IPv6) es `2001:db8:1234:1a00::/64`.

En muchos casos, puede conectarse a través de un proveedor de servicios de internet (ISP) o protegido por un firewall sin una dirección IP estática. Si es así, busque el rango de direcciones IP utilizadas por los ordenadores cliente.

> ⚠ Warning
>
> Si utiliza `0.0.0.0/0` para IPv4 o `::0` para IPv6, permitirá que todas las direcciones IP tengan acceso a las instancias públicas mediante SSH. Este método es aceptable para un periodo de tiempo corto en un entorno de prueba, pero no es seguro en entornos de producción. En los entornos de producción, debe autorizar el acceso a sus instancias únicamente a una dirección IP o a un rango de direcciones IP específicos.

b. En la sección Inbound rules (Reglas de entrada), elija Add rule (agregar regla).

c. Establezca los siguientes valores para la regla de entrada nueva con objeto de permitir el acceso SSH a la instancia de Amazon EC2. Si lo hace, podrá conectarse a la instancia de EC2 para instalar aplicaciones u otras utilidades. Especifique una dirección IP para permitir poder acceder a su instancia de EC2:

- Tipo — **`SSH`**
- Origen: la dirección IP o el rango que creó en el Paso a. Un ejemplo de un rango de direcciones IPv4 es **`203.0.113.25/3`**2. Un ejemplo de un rango de direcciones IPv6 es **`2001:DB8::/32`**.

d. Seleccione Agregar regla.

5. Para crear el grupo de seguridad, elija Create security group (Crear grupo de seguridad).

Anote el identificador del grupo de seguridad, ya que lo necesitará más tarde en otro procedimiento.

Paso 3: crear un grupo de seguridad de VPC para un clúster privado

Para crear un clúster privado, debe crear un segundo grupo de seguridad para el acceso privado. Para conectarse a clústeres privados en su VPC, añada reglas de entrada a su grupo de seguridad de VPC. Estos permiten el tráfico de su instancia de Amazon EC2 solamente.

Para crear un grupo de seguridad de VPC

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.
2. Elija VPC Dashboard (Panel VPC), seguido de Security Groups (Grupos de seguridad) y, por último, Create Security Group (Crear grupo de seguridad).
3. En la página Create Security Group (Crear grupo de seguridad), establezca estos valores:
   - Nombre del grupo de seguridad: **`example-dual-stack-cluster-securitygroup`**
   - Descripción — **`Dual-stack cluster security group`**
   - VPC: elija la VPC que creó anteriormente, por ejemplo: vpc-example-dual-stack.
4. Agregar reglas de entrada al grupo de seguridad
   a. En la sección Inbound rules (Reglas de entrada), elija Add rule (gregar regla).
   b. Establezca los siguientes valores para la regla de entrada nueva a fin de permitir el tráfico de DocumentDB en el puerto 27017 desde la instancia de Amazon EC2. Si lo hace, podrá conectarse desde su instancia de EC2 a su clúster. De esta manera podrá enviar datos desde la instancia de EC2 a la base de datos.
      - Tipo — **`Custom TCP`**
      - Origen: por ejemplo, el identificador del grupo de seguridad de EC2 que creó anteriormente en este tema: sg-9edd5cfb.
5. Para crear el grupo de seguridad, elija Create security group (Crear grupo de seguridad).

Paso 4: Crear un grupo de subredes

Un grupo de subredes es una colección de subredes que se crean en una VPC y que después se asignan a los clústeres. Un grupo de subredes le permite especificar una VPC específica cuando crea clústeres. Para crear un grupo de subredes que sea compatible con `DUAL`, todas las subredes deben ser compatibles con `DUAL`. Para que sea compatible con `DUAL`, una subred debe tener asociado un CIDR IPv6.

Para crear un grupo de subredes

1. Identifique las subredes privadas de la base de datos en la VPC.

   a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.

   b. Seleccione VPC Dashboard (Panel de control de VPC) y, a continuación, seleccione Subnets (Subredes).

   c. Anote los identificadores de subred de las subredes que creó en el Paso 1 denominadas, por ejemplo: example-dual-stack-subnet-private1-us-west-2a y example-dual-stack-subnet-private2-us-west-2b. Necesitará los identificadores de subred cuando cree el grupo de subredes.

2. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb

   Asegúrese de conectarse a la consola de Amazon DocumentDB, no a la consola de Amazon VPC.

3. En el panel de navegación, elija Subnet groups (grupos de subredes).

4. Seleccione Crear.

5. En la página Crear grupo de subredes, establezca estos valores en la sección Detalles del grupo de subredes:

   - Nombre — **`example-dual-stack-cluster-subnet-group`**
   - Descripción — **`Dual-stack cluster subnet group`**

6. En la sección Configuración, establezca los siguientes valores:

   - VPC: elija la VPC que creó anteriormente, por ejemplo: vpc-example-dual-stack.
   - Zonas de disponibilidad: seleccione las dos zonas de disponibilidad creadas en el Paso 1. Ejemplo: us-west-2a y us-west-2b.
   - Subredes: elija las subredes privadas que creó en el Paso 1.

7. Seleccione Crear.

El nuevo grupo de subredes aparece en la lista de grupos de subredes de la consola de DocumentDB. Puede elegir el grupo de subredes para ver los detalles en el panel de detalles. Estos detalles incluyen todas las subredes asociadas al grupo.

Paso 5: crear una instancia de Amazon EC2 en el modo de pila doble

Para crear una instancia de Amazon EC2, siga las instrucciones indicadas en Inicialización de una instancia de EC2 mediante el asistente de inicialización de instancias de la consola en la Guía del usuario de Amazon Elastic Compute Cloud.

En la página Configure Instance Details (Configurar detalles de instancia), defina estos valores y mantenga los demás con sus valores predeterminados:

- Red: elija una VPC existente con las subredes pública y privada, como vpc-example-dual-stack-vpc (vpc-identifier) creada en Paso 1: crear una VPC con subredes públicas y privadas.
- Subred: elija una subred pública existente, como subnet-identifier | example-dual-stack-subnet-public1-us-east-2a | us-east-2a creada en Paso 2: crear un grupo de seguridad de la VPC para una instancia de Amazon EC2 pública.
- Auto-assign IP pública: seleccione Activar.
- Auto-assign IP IPv6: elija Habilitar.
- Firewall (grupos de seguridad): elija Seleccionar un grupo de seguridad existente.
- Grupos de seguridad comunes: elija un grupo de seguridad existente, como example-dual-stack-securitygroup creado en Paso 2: crear un grupo de seguridad de la VPC para una instancia de Amazon EC2 pública. Asegúrese de que el grupo de seguridad que elija incluya reglas de entrada para acceso Secure Shell (SSH) y HTTP.

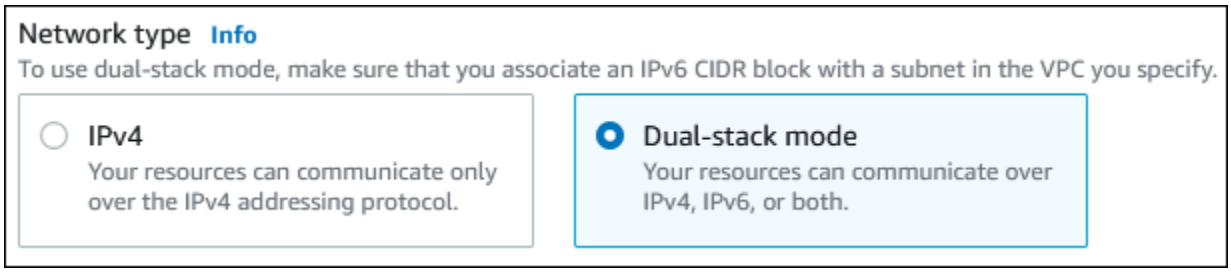
Paso 6: crear un clúster en modo de pila doble

En este paso, debe crear un clúster de base de datos que se ejecute en modo de pila doble. Nota importante: Esta sección debe editarse una vez que se hayan realizado las actualizaciones de IPv6 en la consola.

Cómo crear un clúster en modo de pila doble

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En la esquina superior derecha de la consola, elija el Región de AWS lugar en el que desea crear el clúster de DocumentDB. En este ejemplo se utiliza la región Este de EE. UU. (Ohio).
3. En el panel de navegación, seleccione Clusters (Clústeres).
4. En la página de lista de Clústeres, elija Crear.

5. En la página Crear un clúster de Amazon DocumentDB, asegúrese de elegir la opción de Instance-based clúster.
6. En la sección Conectividad, en Tipo de red, elija Dual-stack el modo.



7. En la parte inferior de la página, active Mostrar configuración avanzada.
8. En la sección Configuración de la red, establezca los siguientes valores:

   - Nube privada virtual (VPC): elija una VPC existente con las subredes pública y privada, como vpc-example-dual-stack (vpc-identifier) creada en Paso 1: crear una VPC con subredes públicas y privadas.

     La VPC debe tener subredes en diferentes zonas de disponibilidad.
   - Grupo de subredes: grupo de subredes para la VPC, como example-dual-stack-cluster-subnet-group creado en Paso 4: Crear un grupo de subredes.
   - Acceso público: elija No.
   - Grupo de seguridad de VPC (firewall): seleccione Elegir existente.
   - Grupos de seguridad de VPC existentes: elija un grupo de seguridad de la VPC existente configurado para el acceso público, como example-dual-stack-cluster-securitygroup creado en Paso 3: crear un grupo de seguridad de VPC para un clúster privado.

     Elimine otros grupos de seguridad, como el grupo de seguridad predeterminado, seleccionando la X asociada con cada uno de ellos.
   - Zonas de disponibilidad: seleccione la zona de disponibilidad creada en el Paso 1. Ejemplo: us-west-2a.

     Para evitar el tráfico entre zonas de disponibilidad, asegúrese de que el clúster y la instancia de EC2 estén en la misma zona de disponibilidad.
9. En el resto de las secciones, especifique la configuración de su clúster. Para obtener más información acerca de cada configuración, consulte Creación de un clúster de Amazon DocumentDB.

Paso 7: conectarse a la instancia de Amazon EC2 y al clúster de base de datos

Después de crear la instancia de Amazon EC2 y el clúster de DocumentDB en el modo de pila doble, puede conectarse a cada una mediante el protocolo IPv6. Para conectarse a una instancia de Amazon EC2 mediante el protocolo IPv6, siga las instrucciones indicadas en Conexión con instancias EC2 en la Guía del usuario de Amazon Elastic Compute Cloud.

Para conectarse al clúster de DocumentDB desde la instancia de EC2, siga las instrucciones en Paso 5: instalación del intérprete de comandos de MongoDB del tema Conexión manual de Amazon EC2 (y continúe con los Pasos 6 y 7 siguientes del mismo procedimiento).

Elimine la VPC

Puede eliminar una VPC y los demás recursos que se utilizan en ella si ya no son necesarios.

> **Note**
>
> Si agregó recursos en la VPC que creó en este tema, es posible que primero tenga que eliminar estos para poder eliminar la VPC. Por ejemplo, estos recursos pueden incluir instancias de Amazon EC2 o clústeres de DocumentDB. Para obtener más información, consulte Eliminación de la VPC en la Guía del usuario de Amazon VPC.

Para eliminar una VPC y los recursos relacionados

1. Elimine el grupo de subredes:
    a. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
    b. En el panel de navegación, elija Subnet groups (grupos de subredes).
    c. Seleccione el grupo de subredes que desea eliminar, por ejemplo, example-dual-stack-cluster-subnet-group.
    d. Elija Delete (Eliminar) y, a continuación, elija Delete (Eliminar) en la ventana de confirmación.
2. Anote el ID de la VPC:
    a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.
    b. Seleccione Panel de VPC y, a continuación, seleccione Sus VPC.

c. En la lista, identifique la VPC que creó, por ejemplo, vpc-example-dual-stack.

d. Anote el ID de la VPC que ha creado. Necesitará el ID de la VPC en pasos posteriores.

3. Elimine los grupos de seguridad:

a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.

b. Seleccione VPC Dashboard (Panel de control de VPC) y, a continuación, seleccione Security Groups (Grupos de seguridad).

c. Seleccione el grupo de seguridad del clúster de Amazon DocumentDB, por ejemplo, example-dual-stack-securitygroup.

d. En Acciones, elija Eliminar grupos de seguridad y, a continuación, seleccione Eliminar en el diálogo de confirmación.

e. En la página Grupos de seguridad, seleccione el grupo de seguridad que corresponda a la instancia de Amazon EC2, por ejemplo, example-securitygroup.

f. En Acciones, elija Eliminar grupos de seguridad y, a continuación, seleccione Eliminar en el diálogo de confirmación.

4. Elimine la puerta de enlace NAT:

a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.

b. Seleccione VPC Dashboard (Panel de control de VPC) y, a continuación, seleccione Security Groups (Grupos de seguridad).

c. Seleccione la puerta de enlace NAT de la VPC que creó. Utilice el ID de VPC para identificar la puerta de enlace NAT correcta.

d. En Actions (Acciones), seleccione Delete NAT gateway (Eliminar puerta de enlace NAT).

e. En el diálogo de confirmación, introduzca **`delete`** y, a continuación, elija Eliminar.

5. Elimine la VPC:

a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.

b. Seleccione Panel de VPC y, a continuación, seleccione Sus VPC.

c. Seleccione la VPC que desea eliminar, por ejemplo, vpc-example-dual-stack.

d. En Acciones, elija Eliminar VPC.

La página de confirmación muestra otros recursos asociados a la VPC que también se eliminarán, incluidas las subredes asociadas a ella.

e. En el diálogo de confirmación, introduzca **`delete`** y, a continuación, elija Eliminar.

6. Lance la dirección IP elástica.

   a. Abra la consola de EC2 en https://console.aws.amazon.com/ec2.

   b. Seleccione EC2 Dashboard (Panel de EC2) y, a continuación, seleccione Elastic IPs (Direcciones IP elásticas).

   c. Seleccione la dirección IP elástica que desea liberar.

   d. Desde Actions (Acciones), elija Release Elastic IP addresses (Liberar direcciones IP elásticas).

   e. En el cuadro de diálogo de confirmación, elija Lanzar.

Copia de seguridad y restauración en Amazon DocumentDB

Amazon DocumentDB (compatible con MongoDB) crea copias de seguridad de datos de forma continua en Amazon Simple Storage Service (Amazon S3) durante un periodo de 1 a 35 días para que se pueda restaurar con rapidez a cualquier punto dentro del periodo de retención de copia de seguridad. Amazon DocumentDB también toma instantáneas automáticas de los datos como parte de este proceso de copia de seguridad continua.

> **Note**
>
> Se trata de buckets de Amazon S3 administrados por servicios, por lo que usted no tendrá acceso a los archivos de copia de seguridad. Para controlar sus propias copias de seguridad, consulte Volcado, restauración, importación y exportación de datos.

También puede conservar los datos de las copias de seguridad después del periodo de retención de copia de seguridad creando una instantánea manual de los datos del clúster. El proceso de copia de seguridad no afecta al rendimiento del clúster.

En esta sección se explican los casos de uso de las funciones de copia de seguridad de Amazon DocumentDB y se muestra cómo administrar las copias de seguridad de los clústeres de Amazon DocumentDB.

Temas

- Conceptos de copia de seguridad y restauración
- Descripción del uso de almacenamiento de copias de seguridad en
- Volcado, restauración, importación y exportación de datos
- Consideraciones sobre las instantáneas de clústeres
- Diferencias entre las instantáneas automáticas y manuales
- Creación de una instantánea manual del clúster
- Copia de instantáneas de clústeres de Amazon DocumentDB
- Uso compartido de instantáneas de clústeres de Amazon DocumentDB
- Restauración de una instantánea del clúster
- Restauración a un momento dado

- Eliminación de una instantánea del clúster

Conceptos de copia de seguridad y restauración

Nombre	Description (Descripción)	API (Verbos)

Backup retention period (Periodo de retención de copia de seguridad)	Periodo de tiempo comprendido entre 1 y 35 días durante el que puede realizar una restauración a un momento dado.	`create-db-cluster`
		`modify-db-cluster`
		`restore-db-cluster-to-point-in-time`

Volumen de almacenamiento de Amazon DocumentDB	Volumen de almacenamiento de alta disponibilidad y larga duración que replica los datos de seis formas en tres	`create-db-cluster` `delete-db-cluster`

Nombre	Description (Descripción)	API (Verbos)
	zonas de disponibilidad. Los clústeres de Amazon DocumentDB son de larga duración independientemente del número de instancias en el clúster.	

Intervalo de copia de seguridad	Periodo de tiempo en el día en el que se realizan las instantáneas automáticas.	`create-db-cluster`
		`describe-db-cluster`
		`modify-db-cluster`

Nombre	Description (Descripción)	API (Verbos)

Instantánea automática	Instantáneas diarias que son copias de seguridad completas del clúster y que se crean automáticamente mediante el proceso de copia de seguridad continua de Amazon DocumentDB.	`restore-db-cluster-from-snapshot`
		
		`describe-db-cluster-snapshot-attributes`
		
		`describe-db-cluster-snapshots`

Nombre	Description (Descripción)	API (Verbos)

Instantánea manual	Instantáneas que crea manualmente para conservar copias de seguridad completas de un clúster más allá del periodo de copia de seguridad.	`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`

Descripción del uso de almacenamiento de copias de seguridad en

El almacenamiento de copias de seguridad de Amazon DocumentDB se compone de copias de seguridad continuas durante el periodo de retención de copia de seguridad y de instantáneas manuales fuera del periodo de retención. Para controlar el uso de almacenamiento de copias de seguridad, puede reducir el intervalo de retención de copia de seguridad, eliminar las instantáneas manuales anteriores cuando ya no las necesite o ambas cosas. Para obtener información general acerca de las copias de seguridad de Amazon DocumentDB, consulte Copia de seguridad y restauración en Amazon DocumentDB. Para obtener información acerca de los precios del almacenamiento de copias de seguridad de Amazon DocumentDB, consulte la página web Precios de Amazon DocumentDB.

Para controlar los costos, puede monitorizar la cantidad de almacenamiento consumido por las copias de seguridad continuas y las instantáneas manuales que persistan más allá del periodo de retención. A continuación, puede reducir el intervalo de retención de copia de seguridad y eliminar las instantáneas manuales cuando ya no las necesite.

Puede utilizar las CloudWatch métricas `TotalBackupStorageBilled` de Amazon y `BackupRetentionPeriodStorageUsed` revisar y supervisar la cantidad de almacenamiento que utilizan las copias de seguridad de Amazon DocumentDB de la siguiente manera: `SnapshotStorageUsed`

- `BackupRetentionPeriodStorageUsed` representa la cantidad de almacenamiento de copias de seguridad utilizado para almacenar copias de seguridad continuas en el momento actual. El valor de esta métrica depende del tamaño del volumen del clúster y de la cantidad de cambios que realice durante el periodo de retención. Sin embargo, a efectos de facturación, la métrica no supera el tamaño del volumen del clúster acumulado durante el periodo de retención. Por ejemplo, si el tamaño del clúster es de 100 GiB y el periodo de retención es de dos días, el valor máximo de `BackRetentionPeriodStorageUsed` es 200 GiB (100 GiB + 100 GiB).

- `SnapshotStorageUsed` representa la cantidad de almacenamiento de copias de seguridad utilizado para almacenar instantáneas manuales más allá del periodo de retención de copia de seguridad. Las instantáneas manuales realizadas en el periodo de retención no cuentan para el almacenamiento de copias de seguridad. Asimismo, las instantáneas automáticas tampoco cuentan para el almacenamiento de copias de seguridad. El tamaño de cada instantánea es el tamaño del volumen del clúster en el momento en que se realiza la instantánea. El valor de `SnapshotStorageUsed` depende del número de instantáneas que mantiene y del tamaño de cada instantánea. Suponga, por ejemplo, que tiene una instantánea fuera del periodo de retención y que el volumen del clúster tenía un tamaño de 100 GiB cuando se realizó la instantánea. La cantidad de `SnapshotStorageUsed` es 100 GiB.

- `TotalBackupStorageBilled` representa la suma de `BackupRetentionPeriodStorageUsed` y `SnapshotStorageUsed`, menos una cantidad de almacenamiento gratuita de copias de seguridad equivalente al tamaño del volumen de un clúster de un día. Por ejemplo, si el tamaño del clúster es de 100 GiB, tiene un día de retención y hay una instantánea fuera del periodo de retención, `TotalBackupStorageBilled` será 100 GiB (100 GiB + 100 GiB - 100 GiB).

- Estas métricas se calculan de forma independiente para cada clúster de Amazon DocumentDB.

Puede supervisar los clústeres de Amazon DocumentDB y crear informes utilizando CloudWatch métricas a través de la CloudWatch consola. Para obtener más información sobre cómo usar CloudWatch las métricas, consulteSupervisión y registro en Amazon DocumentDB.

Volcado, restauración, importación y exportación de datos

Puede utilizar las utilidades `mongodump`, `mongorestore`, `mongoexport` y `mongoimport` para mover datos dentro y fuera del clúster de Amazon DocumentDB. En esta sección se analiza la finalidad de cada una de estas herramientas y configuraciones para ayudarle a lograr un mejor rendimiento.

> **Requisito de versión de MongoDB Database Tools**
>
> Utilice las herramientas de base de datos de MongoDB hasta la versión 100.11.0, incluida, para Amazon DocumentDB. Para descargarlas, consulte las versiones de las herramientas de base de datos de MongoDB en el sitio web de MongoDB.

Temas

- mongodump
- mongorestore
- mongoexport
- importación de mongo
- Tutorial

`mongodump`

La utilidad `mongodump` crea una copia de seguridad binaria (BSON) de una base de datos MongoDB. La herramienta `mongodump` es el método preferido para volcar datos de su implementación de MongoDB de origen cuando busca restaurarla en su clúster de Amazon DocumentDB debido a las eficiencias de tamaño logradas al almacenar los datos en un formato binario.

En función de los recursos disponibles en la instancia o el equipo que esté utilizando para ejecutar el comando, puede acelerar su `mongodump` aumentando el número de colecciones paralelas volcadas desde el valor predeterminado 1 utilizando la opción `--numParallelCollections`. Una buena

sugerencia es comenzar con un proceso de trabajo por vCPU en la instancia principal del clúster de Amazon DocumentDB.

Ejemplo de uso

A continuación se muestra un ejemplo de uso de la utilidad mongodump en el clúster de 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

La utilidad `mongorestore` permite restaurar una copia de seguridad binaria (BSON) de una base de datos creada con la utilidad mongodump. Puede mejorar el rendimiento de la restauración aumentando el número de procesos de trabajo para cada colección durante la restauración con la opción `--numInsertionWorkersPerCollection` (el valor predeterminado es 1). Una buena sugerencia es comenzar con un proceso de trabajo por vCPU en la instancia principal del clúster de Amazon DocumentDB.

Ejemplo de uso

A continuación se muestra un ejemplo de uso de la utilidad `mongorestore` en el clúster de 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

La herramienta `mongoexport` exporta datos en Amazon DocumentDB a formatos de archivo JSON, CSV o TSV. La herramienta `mongoexport` es el método preferido para exportar datos que deben ser legibles para las personas o máquinas.

> **Note**
>
> `mongoexport` no admite directamente las exportaciones paralelas. Sin embargo, es posible aumentar el rendimiento ejecutando varios trabajos `mongoexport` simultáneamente para distintas colecciones.

Ejemplo de uso

A continuación se muestra un ejemplo de uso de la herramienta `mongoexport` en el clúster de 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
```

importación de mongo

La herramienta `mongoimport` importa el contenido de los archivos JSON, CSV o TSV en un clúster de Amazon DocumentDB. Puede utilizar el parámetro `--numInsertionWorkers` para paralelizar y acelerar la importación (el valor predeterminado es 1).

Ejemplo de uso

A continuación se muestra un ejemplo de uso de la herramienta `mongoimport` en el clúster de 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

En el siguiente tutorial se describe cómo utilizar las utilidades `mongodump`, `mongorestore`, `mongoexport` y `mongoimport` para mover datos dentro y fuera de un clúster de Amazon DocumentDB.

1. Requisitos previos: antes de empezar, asegúrese de que el clúster de Amazon DocumentDB está aprovisionado y de que tiene acceso a una instancia de Amazon EC2 en la misma VPC que el clúster. Para obtener más información, consulte Conectarse mediante Amazon EC2.

   Para poder utilizar las herramientas de utilidad mongo, debe tener instalado el paquete mongodb-org-tools en su instancia EC2, como se indica a continuación.

   ```
   sudo yum install mongodb-org-tools-4.0.18
   ```

   Dado que Amazon DocumentDB utiliza el cifrado de seguridad de la capa de transporte (TLS) de forma predeterminada, también debe descargar el archivo de la autoridad de certificación (CA) de Amazon RDS para utilizar el intérprete de comandos de mongo para conectarse, como se indica a continuación.

   ```
   wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
   ```

2. Descarga de datos de muestra: para este tutorial, descargará algunos datos de muestra que contienen información sobre restaurantes.

   ```
   wget https://raw.githubusercontent.com/ozlerhakan/mongodb-json-files/master/
   datasets/restaurant.json
   ```

3. Importación de los datos de muestra a Amazon DocumentDB: dado que los datos están en un formato JSON lógico, utilizará la utilidad `mongoimport` para importar los datos a su clúster de 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. Volcado de los datos con **`mongodump`**: ahora que tiene datos en su clúster de Amazon DocumentDB, puede realizar un volcado binario de esos datos mediante la utilidad `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. Eliminación de la colección **`restaurants`**: antes de restaurar la colección `restaurants` en la base de datos `business`, primero debe eliminar la colección que ya existe en esa base de datos, de la siguiente manera.

```
use business
```

```
db.restaurants.drop()
```

6. Restauración de los datos con **`mongorestore`**: con el volcado binario de los datos del paso 3, ahora puede usar la utilidad `mongorestore` para restaurar los datos en su clúster de 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. Exportación de los datos mediante **`mongoexport`**: para completar el tutorial, exporte los datos del clúster en el formato de un archivo JSON, igual que con el archivo que importó en el paso 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. Validación: puede validar que el resultado del paso 5 genera el mismo resultado que el del paso 1 con los siguientes comandos.

```
wc -l restaurant.json
```

Salida de este comando:

```
2548 restaurant.json
```

```
wc -l restaurant2.json
```

Salida de este comando:

```
2548 restaurant2.json
```

Consideraciones sobre las instantáneas de clústeres

Amazon DocumentDB crea instantáneas automáticas diarias del clúster durante el período de copia de seguridad del clúster. Amazon DocumentDB guarda las instantáneas automáticas del clúster en función del periodo de retención de copia de seguridad que haya especificado. Si es necesario, puede restaurar el clúster a cualquier momento dado durante el período de retención de copia de seguridad. Las instantáneas automáticas no se producen mientras una copia se está ejecutando en la misma región del mismo clúster.

Temas

- Almacenamiento de copia de seguridad
- Intervalo de copia de seguridad
- Backup retention period (Periodo de retención de copia de seguridad)
- Copia del cifrado de instantáneas de clúster

Además de instantáneas automáticas del clúster, también puede crear manualmente una instantánea del clúster. Puede copiar tanto las instantáneas automáticas como las manuales. Para obtener más información, consulte Creación de una instantánea manual del clúster y Copia de instantáneas de clústeres de Amazon DocumentDB.

> **Note**
>
> El clúster debe estar en estado disponible para que se pueda realizar una instantánea automática.
> No se pueden compartir instantáneas automatizadas de un clúster de Amazon DocumentDB. Como solución alternativa, puede crear una instantánea manual copiando la instantánea automatizada y compartir después esa copia. Para obtener más información acerca de cómo copiar una instantánea, consulte Copia de instantáneas de clústeres de Amazon DocumentDB. Para obtener más información acerca de cómo restaurar un clúster desde una instantánea, consulte Restauración de una instantánea del clúster.

Almacenamiento de copia de seguridad

El almacenamiento de copias de seguridad de Amazon DocumentDB para cada una de ellas Región de AWS está compuesto por el almacenamiento de copias de seguridad necesario para el período de retención de las copias de seguridad, que incluye las instantáneas automáticas y manuales de los clústeres en esa región. El período predeterminado de retención de copia de seguridad es de un día. Para obtener más información acerca de los precios del almacenamiento de copias de seguridad, consulte Precios de Amazon DocumentDB.

Al eliminar un clúster, se eliminan todas las instantáneas automáticas y no se pueden recuperar. Sin embargo, las instantáneas manuales no se eliminan al eliminar un clúster. Si decide hacer que Amazon DocumentDB cree una instantánea final (instantánea manual) antes de eliminar el clúster, puede utilizar la instantánea final para recuperar el clúster.

Para obtener más información acerca de las instantáneas y el almacenamiento, consulte Descripción del uso de almacenamiento de copias de seguridad en.

Intervalo de copia de seguridad

Las instantáneas automáticas se realizan a diario durante el periodo de copia de seguridad preferido. Si la instantánea requiere más tiempo del asignado al periodo de copia de seguridad, el proceso de copia de seguridad continúa hasta que finaliza, aunque haya terminado el periodo de copia de seguridad. El periodo de copia de seguridad no se puede solapar con el periodo de mantenimiento semanal del clúster.

Si no especifica un periodo de copia de seguridad preferido al crear el clúster, Amazon DocumentDB asigna un período de copia de seguridad predeterminado de 30 minutos. Este periodo se elige al azar de un bloque de tiempo de 8 horas asociado a la región de su clúster. Puede cambiar el periodo de copia de seguridad preferido modificando el clúster. Para obtener más información, consulte Modificación de un clúster de Amazon DocumentDB.

Nombre de la región	Region	Bloque de tiempo en UTC
Este de EE. UU. (Ohio)	us-east-2	03:00-11:00
Este de EE. UU. (Norte de Virginia)	us-east-1	03:00-11:00
Oeste de EE. UU. (Oregón)	us-west-2	06:00-14:00
África (Ciudad del Cabo)	af-south-1	03:00-11:00
Asia Pacífico (Hong Kong)	ap-east-1	06:00-14:00
Asia-Pacífico (Hyderabad)	ap-south-2	06:30–14:30
Asia-Pacífico (Malasia)	ap-southeast-5	13:00-21:00
Asia-Pacífico (Mumbai)	ap-south-1	06:00-14:00
Asia-Pacífico (Osaka)	ap-northeast-3	12:00–20:00
Asia-Pacífico (Seúl)	ap-northeast-2	13:00-21:00

Nombre de la región	Region	Bloque de tiempo en UTC
Asia-Pacífico (Singapur)	ap-southeast-1	14:00–22:00
Asia-Pacífico (Sídney)	ap-southeast-2	12:00–20:00
Asia-Pacífico (Yakarta)	ap-southeast-3	08:00-16:00
Asia-Pacífico (Melbourne)	ap-southeast-4	11:00-19:00
Asia-Pacífico (Tailandia)	ap-southeast-7	15:00-23:00
Asia-Pacífico (Tokio)	ap-northeast-1	13:00-21:00
Canadá (centro)	ca-central-1	03:00-11:00
Oeste de Canadá (Calgary)	ca-west-1	18:00-02:00
China (Pekín)	cn-north-1	06:00-14:00
China (Ningxia)	cn-northwest-1	06:00-14:00
Europa (Fráncfort)	eu-central-1	21:00-05:00
Europa (Zúrich)	eu-central-2	02:00-10:00
Europa (Irlanda)	eu-west-1	22:00-06:00
Europa (Londres)	eu-west-2	22:00-06:00
Europa (Milán)	eu-south-1	02:00-10:00
Europa (París)	eu-west-3	23:59-07:29
Europa (España)	eu-south-2	02:00-10:00
Europa (Estocolmo)	eu-north-1	04:00 — 12:00
México (centro)	mx-central-1	03:00-11:00
Medio Oriente (EAU)	me-central-1	05:00-13:00

Nombre de la región	Region	Bloque de tiempo en UTC
América del Sur (São Paulo)	sa-east-1	00:00-08:00
Israel (Tel Aviv)	il-central-1	04:00-12:00
AWS GovCloud (US-East)	us-gov-east-1	17:00-01:00
AWS GovCloud (US-West)	us-gov-west-1	06:00-14:00

Backup retention period (Periodo de retención de copia de seguridad)

El periodo de retención de copia de seguridad es el número de días que se conserva una copia de seguridad automática antes de que se elimine automáticamente. Amazon DocumentDB admite un periodo de retención de copia de seguridad de 1 a 35 días.

Puede configurar el periodo de retención de copia de seguridad al crear el clúster. Si no configura explícitamente el periodo de retención de copia de seguridad, se asigna al clúster el valor predeterminado de un día. Tras crear un clúster, puede modificar el período de retención de las copias de seguridad modificando el clúster mediante el Consola de administración de AWS o el AWS CLI. Para obtener más información, consulte Modificación de un clúster de Amazon DocumentDB.

Copia del cifrado de instantáneas de clúster

El cifrado de clústeres e instantáneas se basa en una clave de cifrado de KMS. El ID de la clave de KMS es el Nombre de recurso de Amazon (ARN), el identificador de la clave de KMS o el alias de la clave de KMS de la clave de cifrado de KMS.

Se aplican las siguientes directrices y limitaciones:

- El cifrado se deduce del clúster al crear una instantánea. Si el clúster está cifrado, la instantánea de ese clúster se cifra con la misma clave de KMS. Si el clúster no está cifrado, la instantánea no estará cifrada.
- Si copia una instantánea del clúster cifrada desde la cuenta de Amazon Web Services, puede especificar un valor para `KmsKeyId` para cifrar la copia con una nueva clave de cifrado de KMS. Si no especifica ningún valor para `KmsKeyId`, la copia de la instantánea del clúster se cifra con la misma clave de KMS que la instantánea del clúster de origen.

- Si copia una instantánea del clúster cifrada que se ha compartido desde otra cuenta de Amazon Web Services, debe especificar un valor para `KmsKeyId`.
- Para copiar una instantánea de clúster cifrada a otra región de Amazon Web Services, establezca `KmsKeyId` al ID de clave de KMS que desee utilizar para cifrar la copia de la instantánea de clúster en la región de destino. Las claves de cifrado de KMS son específicas de la región de Amazon Web Services en la que se han creado, y no puede usar claves de cifrado de una región de Amazon Web Services en otra región de Amazon Web Services.
- Si intenta copiar un snapshot de clúster de base de datos sin cifrar y especificar un valor para el parámetro `KmsKeyId`, se devuelve un error.

Diferencias entre las instantáneas automáticas y manuales

A continuación se detallan las principales características de las instantáneas automáticas y manuales de Amazon DocumentDB (compatible con MongoDB).

Las instantáneas automáticas de Amazon DocumentDB tienen las siguientes principales características:

- Nomenclatura de las instantáneas automáticas: los nombres de las instantáneas automáticas siguen el patrón `rds:<cluster-name>-yyyy-mm-dd-hh-mm`, donde `yyyy-mm-dd-hh-mm` representa la fecha y la hora en la que se creó la instantánea.
- Se crean automáticamente según una programación: al crear o modificar un clúster, puede definir el periodo de retención de copia de seguridad como un valor entero comprendido entre 1 y 35 días. De forma predeterminada, los clústeres nuevos tienen un periodo de retención de copia de seguridad de un día. El periodo de retención de copia de seguridad define el número de días que se conservan las instantáneas automáticas antes de que se eliminen automáticamente. No se pueden deshabilitar las copias de seguridad automáticas en los clústeres de Amazon DocumentDB.

  Además de definir el periodo de retención de copia de seguridad, también configura el periodo de copia de seguridad, que es la hora del día a la que se crean las instantáneas automáticas.
- Eliminación de instantáneas automáticas: las instantáneas automáticas se eliminan cuando elimina el clúster de la instantánea automática. No se puede eliminar manualmente una instantánea automática.

- Son incrementales: durante el periodo de retención de copia de seguridad, se registran las actualizaciones de la base de datos, por lo que se mantiene un registro incremental de los cambios.
- Restauración desde una instantánea automática: puede restaurar los datos desde una instantánea automática mediante la Consola de administración de AWS o la AWS CLI. Cuando restaures desde una instantánea con el AWS CLI, debes agregar instancias por separado una vez que el clúster esté disponible.
- Uso compartido: no se puede compartir una instantánea de clúster automatizada de Amazon DocumentDB. Como solución alternativa, puede crear una instantánea manual copiando la instantánea automatizada y compartir después esa copia. Para obtener más información acerca de cómo copiar una instantánea, consulte Copia de instantáneas de clústeres de Amazon DocumentDB. Para obtener más información acerca de cómo restaurar un clúster desde una instantánea, consulte Restauración de una instantánea del clúster.
- Puede restaurar desde cualquier punto durante el periodo de retención de copia de seguridad: como las actualizaciones de la base de datos se registran de manera incremental, puede restaurar el clúster a cualquier punto en el tiempo durante el periodo de retención de copia de seguridad.

  Cuando restauras desde una instantánea automática o desde una restauración puntual con la AWS CLI, debes agregar instancias por separado una vez que el clúster esté disponible.

Las instantáneas manuales de Amazon DocumentDB tienen las siguientes características principales:

- Creadas bajo demanda: las instantáneas manuales de Amazon DocumentDB se crean a pedido mediante la consola de administración de Amazon DocumentDB o. AWS CLI
- Eliminación de una instantánea manual: una instantánea manual solo se elimina cuando se elimina de forma explícita mediante la consola de Amazon DocumentDB o la AWS CLI. Las instantáneas manuales no se eliminan cuando se elimina el clúster.
- Copias de seguridad completas: cuando se realiza una instantánea manual, se crea y se almacena una copia de seguridad completa de los datos del clúster.
- Nomenclatura de instantáneas manuales: usted especifica el nombre de la instantánea manual. Amazon DocumentDB no añade una marca `datetime` al nombre, por lo que debe añadir esa información si desea que se incluya en el nombre.

- Restauración desde una instantánea manual: puede restaurar los datos desde una instantánea manual mediante la consola o la AWS CLI. Cuando restaure a partir de una instantánea utilizando la AWS CLI, debe agregar instancias por separado una vez que el clúster esté disponible.
- Cuotas de servicio: está limitado a un máximo de 100 instantáneas manuales por cada Región de AWS una.
- Uso compartido: puede compartir instantáneas de clúster manuales, que las Cuentas de AWS autorizadas pueden copiar. Puede compartir instantáneas manuales cifradas o no cifradas. Para obtener más información acerca de cómo copiar una instantánea, consulte Copia de instantáneas de clústeres de Amazon DocumentDB.
- La restauración tiene lugar cuando se realiza la instantánea manual: cuando se restauran los datos desde una instantánea manual, la restauración tiene lugar cuando se realiza la instantánea manual.

  Cuando restaures a partir de una instantánea con el AWS CLI, debes agregar instancias por separado una vez que el clúster esté disponible.

Creación de una instantánea manual del clúster

Puede crear una instantánea manual mediante el Consola de administración de AWS o AWS CLI. El tiempo que tarda en crearse una instantánea varía en función del tamaño de sus bases de datos. Cuando cree una instantánea, debe hacer lo siguiente:

1. Identificar el clúster del que va a realizar una copia de seguridad.
2. Asignar un nombre a la instantánea. Esto le permite restaurarla más adelante.

Using the Consola de administración de AWS

Para crear una instantánea manual con el Consola de administración de AWS, puede seguir cualquiera de los métodos siguientes.

1. Método 1:

   1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
   2. En el panel de navegación, elija Snapshots (Instantáneas).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En la página Snapshots (Instantáneas), elija Create (Crear).
4. En la página Create cluster snapshot (Crear instantánea del clúster):
   a. Identificador de clúster: en la lista desplegable de clústeres, elija el clúster del que desea crear una instantánea.
   b. Identificador de la instantánea: introduzca un nombre para la instantánea.

      Restricciones relativas a la nomenclatura de instantáneas:

      - Debe tener [1-255] letras, números o guiones.
      - El primer carácter debe ser una letra.
      - No puede terminar con un guion ni contener dos guiones consecutivos.
      - Debe ser único para todos los clústeres de Amazon RDS, Amazon Neptune y Amazon DocumentDB por cuenta de AWS y por región.

   c. Seleccione Crear.

2. Método 2:

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Clusters (Clústeres).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En la página Clusters (clústeres), elija el botón situado a la izquierda del clúster cuya instantánea desee crear.

4. En el menú Actions (Acciones), elija Take snapshot (Realizar instantánea).
5. En la página Create cluster snapshot (Crear instantánea del clúster):
   a. Identificador de la instantánea: introduzca un nombre para la instantánea.

      Restricciones relativas a la nomenclatura de instantáneas:

      - Debe tener [1-63] letras, números o guiones.
      - El primer carácter debe ser una letra.
      - No puede terminar con un guion ni contener dos guiones consecutivos.
      - Debe ser único para todos los clústeres de Amazon RDS, Amazon Neptune y Amazon DocumentDB por cuenta de AWS y por región.

   b. Seleccione Crear.

Using the AWS CLI

Para crear una instantánea de clúster mediante AWS CLI, utilice la `create-db-cluster-snapshot` operación con los siguientes parámetros.

Parameters

- **`--db-cluster-identifier`**: obligatorio. El nombre del clúster del que va a realizar una instantánea. Este clúster debe existir y estar disponible.
- **`--db-cluster-snapshot-identifier`**: obligatorio. El nombre de la instantánea manual que va a crear.

En el ejemplo siguiente, se crea una instantánea llamada `sample-cluster-snapshot` del clúster `sample-cluster`.

Para Linux, macOS o Unix:

```
aws docdb create-db-cluster-snapshot \
    --db-cluster-identifier sample-cluster \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Para Windows:

```
aws docdb create-db-cluster-snapshot ^
```

```
    --db-cluster-identifier sample-cluster ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "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 de instantáneas de clústeres de Amazon DocumentDB

En Amazon DocumentDB, puede copiar las instantáneas dentro de la misma página Región de AWS o en una diferente. Región de AWS También puede copiar las instantáneas compartidas a su cuenta en la misma cuenta Región de AWS o en otra diferente. Región de AWS Para obtener más información sobre cómo compartir instantáneas, consulte Uso compartido de instantáneas de clústeres de Amazon DocumentDB.

> **Note**
>
> Amazon DocumentDB le factura en función de la cantidad de datos de copias de seguridad e instantáneas que conserve y el periodo de tiempo que los conserve. Para obtener más información sobre el almacenamiento asociado con las copias de seguridad y las instantáneas de Amazon DocumentDB, consulte Descripción del uso de almacenamiento de copias de seguridad en. Para obtener información acerca de los precios del almacenamiento de Amazon DocumentDB, consulte Precios de Amazon DocumentDB.

Temas

- Copia de instantáneas compartidas
- Copiar instantáneas de un lado a otro Regiones de AWS
- Limitaciones
- Tratamiento del cifrado
- Consideraciones relativas al grupo de parámetros
- Copia de una instantánea de clúster

Copia de instantáneas compartidas

Puedes copiar las instantáneas que te hayan compartido otras AWS cuentas. Si va a copiar una instantánea cifrada que se ha compartido desde otra AWS cuenta, debe tener acceso a la clave de AWS KMS cifrado que se utilizó para cifrar la instantánea. Para obtener más información, consulte Tratamiento del cifrado.

Copiar instantáneas de un lado a otro Regiones de AWS

Al copiar una instantánea a una Región de AWS que es diferente de la instantánea de origen Región de AWS, cada copia es una instantánea completa. Una copia de la instantánea completa contiene todos los datos y metadatos necesarios para restaurar el clúster de Amazon DocumentDB.

Limitaciones

A continuación se indican algunas limitaciones al copiar instantáneas:

- Si elimina una instantánea de origen antes de que la instantánea de destino esté disponible, la copia de la instantánea puede generar un error. Compruebe que la instantánea de destino tiene el estado `AVAILABLE` antes de eliminar una instantánea de origen.
- Puede tener hasta cinco solicitudes de copia de instantánea en curso en una única región de destino por cuenta.
- Según la información de que se Regiones de AWS trate y la cantidad de datos que se vayan a copiar, la copia de una instantánea entre regiones puede tardar horas en completarse. En algunos casos, puede haber un gran número de solicitudes de copia de instantáneas entre regiones desde una Región de AWS. En estos casos, Amazon DocumentDB puede poner en cola las nuevas solicitudes de copia entre regiones de esa fuente hasta que Región de AWS se completen algunas copias en curso. No se muestra ninguna información de progreso sobre las solicitudes de copia mientras están en la cola. La información de progreso se muestra cuando comienza la copia.

Tratamiento del cifrado

Puede copiar una instantánea que haya sido cifrada con una clave de cifrado de AWS KMS . Si copia una instantánea cifrada, la copia de la instantánea se debe cifrar también. Si copia una instantánea cifrada dentro de la misma Región de AWS, puede cifrar la copia con la misma clave de AWS KMS cifrado que la instantánea original o puede especificar una clave de cifrado diferente. AWS KMS Si copias una instantánea cifrada en varias regiones, no puedes usar la misma clave de AWS KMS cifrado para la copia que para la instantánea de origen, ya que AWS KMS las claves sí lo son Region-specific. En su lugar, debes especificar una AWS KMS clave válida en el destino Región de AWS n.

La instantánea de origen permanece cifrada durante todo el proceso de copia. Para obtener más información, consulte Protección de datos en Amazon DocumentDB.

> **Note**
>
> Para instantáneas del clúster de Amazon DocumentDB, no es posible cifrar una instantánea del clúster sin cifrar al copiar la instantánea.

Consideraciones relativas al grupo de parámetros

Cuando se copia una instantánea entre regiones, la copia no incluye el grupo de parámetros empleado por el clúster de Amazon DocumentDB. Al restaurar una instantánea para crear un clúster

nuevo, ese clúster obtiene el grupo de parámetros predeterminado para el grupo en el Región de AWS que se creó. Para aplicar en el nuevo clúster los mismos parámetros que en el original, debe hacer lo siguiente:

1. En el destino Región de AWS, cree un grupo de parámetros de clúster de Amazon DocumentDB con la misma configuración que el clúster original. Si ya existe uno en el nuevo Región de AWS, puede usarlo.
2. Tras restaurar la instantánea en el destino Región de AWS, modifique el nuevo clúster de Amazon DocumentDB y añada el grupo de parámetros nuevo o existente del paso anterior. Para obtener más información, consulte Modificación de un clúster de Amazon DocumentDB.

Copia de una instantánea de clúster

Puede copiar un clúster de Amazon DocumentDB utilizando el Consola de administración de AWS o el AWS CLI, de la siguiente manera.

Using the Consola de administración de AWS

Para hacer una copia de una instantánea de un clúster mediante el Consola de administración de AWS, complete los pasos siguientes. Este procedimiento funciona para copiar instantáneas de clústeres cifradas o no cifradas, en la misma región Región de AWS o entre regiones.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, elija Instantáneas y, a continuación, el botón situado a la izquierda de la instantánea que desea restaurar.

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En el menú Actions, elija Copy.
4. En la página Hacer una copia de una instantánea de clúster, rellene la sección Configuración.

   a. Región de destino: opcional. Para copiar la instantánea del clúster a otra Región de AWS, selecciónela Región de AWS para la región de destino.

b. Nuevo identificador de instantánea: escriba un nombre para la nueva instantánea.

   Restricciones relativas a la nomenclatura de instantáneas de destino:

   - No se puede coincidir con el nombre de una instantánea existente.
   - Debe tener [1-63] letras, números o guiones.
   - El primer carácter debe ser una letra.
   - No puede terminar con un guion ni contener dos guiones consecutivos.
   - Debe ser único para todos los clústeres de Amazon RDS, Neptune y Amazon DocumentDB por región Cuenta de AWS.

c. Copiar etiquetas: para copiar las etiquetas que tenga en su instantánea de origen a la copia de su instantánea, elija Copiar etiquetas.

5. Complete la sección. Encryption-at-rest

   a. Cifrado en reposo: si la instantánea no está cifrada, estas opciones no estarán disponibles, porque no puede crear una copia sin cifrar a partir de una instantánea cifrada. Si la instantánea está cifrada, puede cambiar la que AWS KMS key se utiliza durante el cifrado en reposo.

      Para obtener más información acerca del cifrado de las copias de instantáneas, consulte Copia del cifrado de instantáneas de clúster.

      Para obtener más información sobre el cifrado en reposo, consulte Cifrado de datos en reposo de Amazon DocumentDB.

   b. AWS KMS Clave: en la lista desplegable, elija una de las siguientes opciones:

      - (predeterminado) aws/rds: el número de cuenta y el ID de la AWS KMS clave aparecen después de esta opción.
      - <some-key-name>: si ha creado una clave, aparecerá y estará disponible para que la seleccione.
      - Introducir un ARN de clave: en el cuadro ARN, escriba el nombre de recurso de Amazon (ARN) de la clave de AWS KMS . El formato del ARN es el siguiente: `arn:aws:kms:<region>:<accountID>:key/<key-id>` .

6. Para crear una copia de la instantánea seleccionada, elija Copy snapshot (Copiar instantánea). También puede seleccionar Cancelar si no desea hacer una copia de la instantánea.

Using the AWS CLI

Para realizar una copia de una instantánea de clúster sin cifrar mediante la AWS CLI, utilice la operación `copy-db-cluster-snapshot` con los siguientes parámetros. Si va a copiar la instantánea a otra Región de AWS, ejecute el comando en el Región de AWS que se copiará la instantánea.

- **`--source-db-cluster-snapshot-identifier`**: obligatorio. El identificador de la instantánea del clúster de la que se va a hacer una copia. Debe existir la instantánea del clúster y tener el estado disponible. Si va a copiar la instantánea a otra Región de AWS o si va a copiar una instantánea de clúster compartida, este identificador debe estar en el formato ARN de la instantánea de clúster de origen. Este parámetro no distingue entre mayúsculas y minúsculas.
- **`--target-db-cluster-snapshot-identifier`**: obligatorio. El identificador de la nueva instantánea del clúster que se va a crear a partir de la instantánea del clúster de origen. Este parámetro no distingue entre mayúsculas y minúsculas.

  Restricciones relativas a la nomenclatura de instantáneas de destino:

  - No se puede coincidir con el nombre de una instantánea existente.
  - Debe tener [1-63] letras, números o guiones.
  - El primer carácter debe ser una letra.
  - No puede terminar con un guion ni contener dos guiones consecutivos.
  - Debe ser único para todos los clústeres de Amazon RDS, Neptune y Amazon DocumentDB por región. Cuenta de AWS
- **`--source-region`**— Si va a copiar la instantánea en otra Región de AWS, especifique la ubicación desde la Región de AWS que se copiará la instantánea del clúster cifrada.

  Si desea copiar la instantánea en otra Región de AWS y no especifica `--source-region`, debe especificar en su lugar la opción `pre-signed-url`. El `pre-signed-url` valor debe ser una URL que contenga una solicitud firmada de la versión 4 para que se invoque la `CopyDBClusterSnapshot` acción en la fuente desde la Región de AWS que se copia la instantánea del clúster. Para obtener más información sobre el`pre-signed-url`, consulte CopyDBClusterSnapshot.
- **`--kms-key-id`**: identificador de la clave de KMS que se va a utilizar para cifrar la copia de la instantánea del clúster.

  Si va a copiar una instantánea de clúster cifrada a otra Región de AWS, este parámetro es obligatorio. Debe especificar una clave de KMS para el destino Región de AWS.

Si va a copiar una instantánea de clúster cifrada en la misma Región de AWS, el parámetro AWS KMS clave es opcional. La copia de la instantánea del clúster se cifra con la misma AWS KMS clave que la instantánea del clúster de origen. Si desea especificar una nueva clave de AWS KMS cifrado para cifrar la copia, puede hacerlo con este parámetro.

- **`--copy-tags`**— Opcional. Las etiquetas y los valores que se van a copiar.

Para cancelar una operación de copia una vez que está en curso, elimine la instantánea del clúster de destino identificado por `--target-db-cluster-snapshot-identifier` o `TargetDBClusterSnapshotIdentifier` mientras esa instantánea del clúster está en el estado copying.

Example

Ejemplo 1: copia de una instantánea sin cifrar a la misma región

En el siguiente AWS CLI ejemplo, se crea una copia de `sample-cluster-snapshot` named `sample-cluster-snapshot-copy` in Región de AWS igual que la instantánea de origen. Cuando se crea la copia, todas las etiquetas de la instantánea original se copian en la copia de la instantánea.

Para 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
```

Para 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
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "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

Ejemplo 2: Copie una instantánea sin cifrar Regiones de AWS

En el siguiente AWS CLI ejemplo, se crea una copia de`sample-cluster-snapshot`, que tiene el ARN`arn:aws:rds:us-east-1:123456789012:cluster-snapshot:sample-cluster-snapshot`. Esta copia recibe el nombre `sample-cluster-snapshot-copy` y está en el lugar Región de AWS en el que se ejecuta el comando.

Para 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
```

Para 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
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "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

Ejemplo 3: Copiar una instantánea cifrada de un lado a otro Regiones de AWS

En el siguiente AWS CLI ejemplo, se crea una copia `sample-cluster-snapshot` de la región us-west-2 a la región us-east-1. Se llama a este comando en la región us-east-1.

Para 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
```

Para 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
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "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

Ejemplo 4: Copie una instantánea compartida sin cifrar en Regiones de AWS

En el siguiente AWS CLI ejemplo, account -123456789012, se crea una copia de una instantánea de clúster sin cifrar `sample-cluster-snapshot` compartida por una cuenta 999999999999 desde la región us-east-1 a la región us-west-2. Se llama a este comando en la región us-west-2. Para obtener más información sobre cómo compartir instantáneas, consulte Uso compartido de una instantánea.

Para 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
```

Para 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
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
 "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

Ejemplo 5: Copie una instantánea compartida cifrada de un lado a otro Regiones de AWS

En el siguiente AWS CLI ejemplo, una cuenta: `123456789012` crea una copia de una instantánea de clúster cifrada `sample-cluster-snapshot` compartida por una cuenta 999999999999 desde la región us-east-1 a la región us-west-2. La instantánea de destino está cifrada con una clave de KMS administrada por el cliente: `arn:aws:kms:us-west-2:123456789012:key/6c1f3264-1797-472b-ba37-03011e682d28`. Se llama a este comando en la región us-west-2. Para obtener más información sobre cómo compartir instantáneas, consulte Uso compartido de una instantánea.

Para 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
```

Para 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
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
 "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**
>
> Para obtener más información acerca del cifrado de las copias de instantáneas, consulte Copia del cifrado de instantáneas de clúster.
> Para obtener más información sobre el cifrado en reposo, consulte Cifrado de datos en reposo de Amazon DocumentDB.

Uso compartido de instantáneas de clústeres de Amazon DocumentDB

Cuando utiliza Amazon DocumentDB, puede compartir una instantánea manual del clúster de las siguientes maneras:

- Compartir una instantánea manual de un clúster, ya sea cifrada o no, permite a AWS las cuentas autorizadas copiar la instantánea.
- Compartir una instantánea manual de un clúster, ya sea cifrada o no, permite a AWS las cuentas autorizadas restaurar directamente un clúster a partir de la instantánea, en lugar de tomar una copia de la misma y restaurarla a partir de esa instantánea.

> **Note**
>
> Para compartir una instantánea automatizada del clúster, cree una instantánea manual del clúster cuando copia la instantánea automatizada y, a continuación, comparta esa copia. Este proceso también se aplica a los recursos generados por la AWS copia de seguridad.

Puede compartir una instantánea manual con hasta 20 personas más. Cuentas de AWS También puede compartir una instantánea manual sin cifrar como pública, lo que hace que esté disponible para todas las cuentas de . Si comparte una instantánea como pública, compruebe que no contiene información privada.

Al compartir instantáneas manuales con otras Cuentas de AWS personas y restaurar un clúster a partir de una instantánea compartida mediante la API de Amazon DocumentDB AWS CLI o la API de Amazon DocumentDB, debe especificar el nombre de recurso de Amazon (ARN) de la instantánea compartida como identificador de la instantánea.

Cómo compartir una instantánea cifrada

Cuando se comparten instantáneas cifradas, se aplican las siguientes restricciones:

- No se pueden compartir instantáneas cifradas como públicas.
- No puede compartir una instantánea que se haya cifrado con la clave de AWS KMS cifrado predeterminada de la cuenta que compartió la instantánea.

Siga estos pasos para compartir instantáneas cifradas.

1. Comparte la clave de cifrado AWS Key Management Service (AWS KMS) que se usó para cifrar la instantánea con todas las cuentas en las que quieras que puedan acceder a la instantánea.

   Para compartir las claves de AWS KMS cifrado con otras AWS cuentas, añádelas a la política de AWS KMS claves. Para obtener más información sobre la actualización de una política de claves, consulte Uso de políticas de clave en AWS KMS en la Guía para AWS Key Management Service desarrolladores. Para ver un ejemplo de cómo crear una política de claves, consulte Creación de una política de IAM para permitir la copia de la instantánea cifrada más adelante en este tema.
2. AWS CLI Utilícela, como se muestra a continuación, para compartir la instantánea cifrada con las demás cuentas.

Permitir el acceso a un AWS KMS clave de cifrado

Para Cuenta de AWS que otra persona pueda copiar una instantánea cifrada compartida desde tu cuenta, la cuenta con la que compartes la instantánea debe tener acceso a la AWS KMS clave que la cifró. Para permitir que otra cuenta acceda a una AWS KMS clave, actualiza la política de claves de la AWS KMS clave con el ARN de la cuenta con la que compartes como principal en la política de AWS KMS claves. A continuación, permita la acción `kms:CreateGrant`.

Después de conceder a una cuenta acceso a tu clave de AWS KMS cifrado, para copiar la instantánea cifrada, esa cuenta debe crear un usuario AWS Identity and Access Management (IAM) si aún no tiene uno. Además, esa cuenta también debe adjuntar una política de IAM a ese usuario de IAM que le permita copiar una instantánea cifrada con tu clave. AWS KMS La cuenta debe ser un usuario de IAM y no puede ser una Cuenta de AWS identidad raíz debido a AWS KMS restricciones de seguridad.

En el siguiente ejemplo de política de claves, el usuario 123451234512 es el propietario de la clave de cifrado. AWS KMS El usuario 123456789012 es la cuenta con la que se comparte la clave. Esta política de claves actualizada permite a la cuenta acceder a la clave. AWS KMS Para ello, incluye el ARN de la Cuenta de AWS identidad raíz del usuario 123456789012 como principal de la política y permite la acción. `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}}
        }
    ]
}
```

Creación de una política de IAM para permitir la copia de la instantánea cifrada

Cuando una persona externa Cuenta de AWS tiene acceso a tu AWS KMS clave, el propietario de esa cuenta puede crear una política que permita a un usuario de IAM que se haya creado para la cuenta copiar una instantánea cifrada con esa clave. AWS KMS

El siguiente ejemplo muestra una política que se puede adjuntar a un usuario de IAM para 123456789012. Cuenta de AWS La política permite al usuario de IAM copiar una instantánea compartida de la cuenta 123451234512 que se haya cifrado con la clave en la región 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
                }
            }
        }
    ]
```

```
}
```

Para obtener información detallada sobre cómo actualizar una política de claves, consulte Uso de políticas de claves en AWS KMS de la Guía para desarrolladores de AWS Key Management Service .

Uso compartido de una instantánea

Puede compartir una instantánea de clúster manual de Amazon DocumentDB (o una copia de una instantánea automatizada) mediante: Consola de administración de AWS AWS CLI

Using the Consola de administración de AWS

Para compartir una instantánea mediante Consola de administración de AWS, siga estos pasos:

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, elija Snapshots (Instantáneas).
3. Seleccione la instantánea manual que desea compartir.
4. En el menú desplegable Acciones, elija Compartir.
5. Elija una de las siguientes opciones para Visibilidad de instantánea de base de datos.

   - Si la fuente no está cifrada, elija Public para permitir que todas las AWS cuentas restauren un clúster a partir de la instantánea manual. O bien, elige Privado para permitir que solo AWS las cuentas que especifiques restauren un clúster a partir de tu instantánea manual.

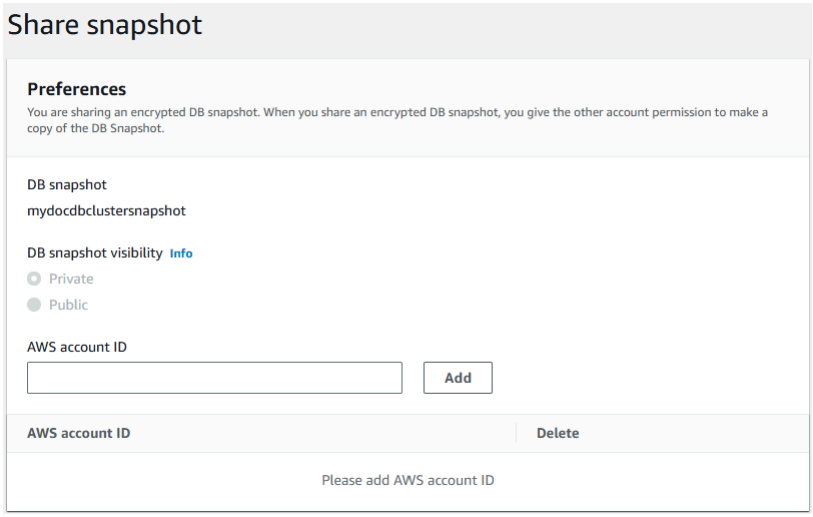
     > **Warning**
     >
     > Si establece la visibilidad de las instantáneas de base de datos en Pública, todas AWS las cuentas pueden restaurar un clúster a partir de la instantánea manual y tener acceso a sus datos. No comparta como Pública ninguna instantánea manual del clúster que contenga información confidencial.

   - Si el original está cifrado, DB Snapshot Visibility (Visibilidad de instantánea de base de datos) se establece en Private (Privada), ya que las instantáneas cifradas no se pueden compartir como públicas.

> **Note**
>
> Las instantáneas que se hayan cifrado con la configuración predeterminada no se AWS KMS key pueden compartir.

6. En el AWS campo ID de AWS cuenta, introduce el identificador de cuenta de una cuenta a la que quieras permitir restaurar un clúster a partir de tu instantánea manual y, a continuación, selecciona Agregar. Repite el procedimiento para incluir identificadores de AWS cuenta adicionales, hasta un máximo de 20 AWS cuentas.

   Si cometes un error al añadir un identificador de AWS cuenta a la lista de cuentas permitidas, puedes eliminarlo de la lista seleccionando Eliminar a la derecha del identificador de AWS cuenta incorrecto.



7. Una vez que haya agregado los identificadores para todas las AWS cuentas a las que desea permitir que restauren la instantánea manual, seleccione Guardar para guardar los cambios.

Using the AWS CLI

Para compartir una instantánea mediante AWS CLI, utilice la operación Amazon DocumentDB. `modify-db-snapshot-attribute` Utilice el `--values-to-add` parámetro para añadir una lista de los ID de las personas Cuentas de AWS que están autorizadas a restaurar la instantánea manual.

El siguiente ejemplo permite que dos Cuenta de AWS identificadores, 123451234512 y 123456789012, restauren la instantánea denominada. `manual-snapshot1` También se elimina el valor del atributo `all` para marcar la instantánea como privada.

Para 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"]'
```

Para Windows:

```
aws docdb modify-db-cluster-snapshot-attribute ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --attribute-name restore ^
    --values-to-add '["123451234512","123456789012"]'
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512",
                    "123456789012"
                ]
            }
        ]
    }
}
```

Para eliminar un Cuenta de AWS identificador de la lista, utilice el parámetro. `--values-to-remove` El siguiente ejemplo evita que el Cuenta de AWS ID 123456789012 restaure la instantánea.

Para 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"]'
```

Para Windows:

```
aws docdb modify-db-cluster-snapshot-attribute ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --attribute-name restore ^
    --values-to-remove '["123456789012"]'
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512"
                ]
            }
        ]
    }
}
```

Restauración de una instantánea del clúster

Amazon DocumentDB (compatible con MongoDB) crea una instantánea del clúster del volumen de almacenamiento. Puede crear un clúster nuevo restaurándolo a partir de una instantánea de clúster. Al restaurar el clúster, debe indicar el nombre de la instantánea del clúster desde la que se realiza la restauración y un nombre para el clúster nuevo que se crea con la operación de restauración. No es posible restaurar desde una instantánea a un clúster existente, ya que al realizar la restauración se crea un clúster nuevo.

Al restaurar un clúster a partir de una instantánea de clúster:

- Esta acción solo restaura el clúster, no las instancias de ese clúster. Debe invocar la acción `create-db-instance` para crear instancias para el clúster restaurado, especificando el identificador de dicho clúster restaurado en `--db-cluster-identifier`. Solo es posible crear las instancias después de que el clúster esté disponible.

- No se puede restaurar una instantánea cifrada en un clúster sin cifrar. Sin embargo, puede restaurar una instantánea sin cifrar en un clúster cifrado especificando la clave. AWS KMS
- Para restaurar un clúster a partir de una instantánea cifrada, debe tener acceso a la AWS KMS clave.

> **Note**
>
> No puede restaurar un clúster 3.6 a uno 4.0, pero puede migrar de una versión de clúster a otra. Para obtener más información, consulte Migración y actualización de Amazon DocumentDB.

Using the Consola de administración de AWS

El procedimiento siguiente muestra cómo restaurar un clúster de Amazon DocumentDB a partir de una instantánea de clúster mediante la consola de administración de Amazon DocumentDB.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, elija Snapshots (Instantáneas) y, a continuación, elija el botón situado a la izquierda de la instantánea que desea utilizar para restaurar un clúster.

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡)
   > en la esquina superior izquierda de la página.

3. En el menú Actions (Acciones), seleccione Restore (Restaurar).
4. En la página Restore snapshot (Restaurar instantánea), rellene la sección Configuration (Configuración).

   a. Identificador de clúster: nombre del nuevo clúster. Puede aceptar el nombre facilitado por Amazon DocumentDB o escribir el nombre que prefiera. El nombre proporcionado por Amazon DocumentDB tiene el formato `docdb-` más una marca de tiempo UTC; por ejemplo, `docdb-yyyy-mm-dd-hh-mm-ss`.

b. Clase de instancia: clase de instancia del nuevo clúster. Puede aceptar la clase de instancia predeterminada o elegir una en la lista desplegable.

c. Número de instancias: número de instancias que desea crear con este clúster. Puede aceptar la configuración predeterminada de 3 instancias (1 réplica principal read/write y 2 réplicas de solo lectura) o elegir la cantidad de instancias en la lista desplegable.

5. En Configuración del almacenamiento en clúster, elija una opción de almacenamiento.

> **Note**
>
> La configuración de I/O-Optimized almacenamiento de Amazon DocumentDB solo está disponible en las versiones 5.0 y 8.0 del motor de Amazon DocumentDB.

6. Si está satisfecho con la configuración del clúster, elija Restore cluster (Restaurar clúster) y espere mientras el clúster se restaura.

7. Si prefiere cambiar algunas configuraciones, como especificar un grupo de seguridad o Amazon VPC no predeterminados, elija Mostrar configuración avanzada en la esquina inferior izquierda de la página y, después, continúe con los pasos siguientes.

   a. Rellene la sección Network settings (Configuración de red).

      - Nube privada virtual (VPC): acepte la VPC actual o elija una de la lista desplegable.
      - Grupo de subredes: acepte el grupo de subredes `default` o elija uno de la lista desplegable.
      - Grupos de seguridad de la VPC: acepte el grupo de seguridad `default (VPC)` o elija uno de la lista.

   b. Rellene la sección Cluster options (Opciones de clúster).

      - Puerto de base de datos: acepte el puerto predeterminado, `27017`, o utilice la flecha hacia arriba o hacia abajo para establecer el puerto que desea utilizar para las conexiones de las aplicaciones.

   c. Rellene la sección Encryption (Cifrado).

      - Cifrado en reposo: si la instantánea está cifrada, estas opciones no están disponibles. Si no está cifrada, puede elegir una de las opciones siguientes:
        - Para cifrar todos los datos del clúster, elija Enable encryption-at-rest (Habilitar cifrado en reposo). Si elige esta opción, debe designar una clave de KMS.

- Si no desea cifrar los datos del clúster, elija Disable encryption-at-rest (Deshabilitar cifrado en reposo). Si elige esta opción, habrá terminado con la sección de cifrado.

- AWS KMS Clave: elija una de las siguientes opciones de la lista desplegable:
  - (predeterminado) aws/rds: el número de cuenta y el ID de la AWS KMS clave aparecen en esta opción.
  - Customer-managed clave: esta opción solo está disponible si ha creado una clave de cifrado de IAM en la consola AWS Identity and Access Management (IAM). Puede elegir la clave que desea utilizar para cifrar el clúster.
  - Introduzca un ARN de clave: en el cuadro ARN, introduzca el nombre de recurso de Amazon (ARN) de su clave. AWS KMS El formato del ARN es el siguiente: `arn:aws:kms:<region>:<accountID>:key/<key-id>`.

d. Rellene la sección Log exports (Exportaciones de registros).

- Seleccione los tipos de registro en los que desea publicar CloudWatch: elija una de las siguientes opciones:
  - Habilitado: permite que el clúster exporte los registros de DDL a Amazon CloudWatch Logs.
  - Desactivado: impide que el clúster exporte los registros de DDL a Amazon CloudWatch Logs. La opción Disabled (Deshabilitado) es la predeterminada.
- Rol de IAM: en la lista, seleccione Rol vinculado al servicio RDS.

e. Complete la sección Tags (Etiquetas).

- Agregar etiqueta: en el cuadro Clave, escriba el nombre de la etiqueta del clúster. En el cuadro Value (Valor), si lo desea, escriba el valor de la etiqueta. Las etiquetas se utilizan en las políticas AWS Identity and Access Management (IAM) para gestionar el acceso a los recursos de Amazon DocumentDB y controlar las acciones que se pueden aplicar a los recursos.

f. Complete la sección Deletion protection (Protección contra la eliminación).

- Habilitar la protección contra la eliminación: protege el clúster para que no se pueda eliminar accidentalmente. Cuando esta opción está habilitada, no se puede eliminar el clúster.

8. Elija Restore cluster (Restaurar clúster).

Using the AWS CLI

Para restaurar un clúster a partir de una instantánea mediante AWS CLI, utilice la `restore-db-cluster-from-snapshot` operación con los siguientes parámetros. Para obtener más información, consulte RestoreDBClusterFromSnapshot.

- **`--db-cluster-identifier`**: obligatorio. Escriba el nombre del clúster que va a crear la operación. No puede existir un clúster con este nombre antes de esta operación.

  Restricciones en cuanto a la nomenclatura de los clústeres:

  - Debe tener [1-63] letras, números o guiones.
  - El primer carácter debe ser una letra.
  - No puede terminar con un guion ni contener dos guiones consecutivos.
  - Debe ser único para todos los clústeres de Amazon RDS, Neptune y Amazon DocumentDB por región Cuenta de AWS.
- **`--snapshot-identifier`**: obligatorio. El nombre de la instantánea desde la que se va a restaurar. Debe existir una instantánea con este nombre y tener el estado disponible.
- **`--engine`**: obligatorio. Debe ser `docdb`.
- **`--storage-type standard | iopt1`**: opcional. Predeterminado: `standard`.
- **`--kms-key-id`**— Opcional. El ARN del identificador de AWS KMS clave que se utilizará al restaurar una instantánea cifrada o al cifrar un clúster al restaurar a partir de una instantánea no cifrada. Al proporcionar el identificador de AWS KMS clave, el clúster restaurado se cifra con la AWS KMS clave, independientemente de que la instantánea esté cifrada o no.

  El formato de `--kms-key-id` es `arn:aws:kms:<region>:<accountID>:key/<key-id>`. Si no se especifica un valor para el parámetro `--kms-key-id`, ocurre lo siguiente:

  - Si la instantánea `--snapshot-identifier` está cifrada, el clúster restaurado se cifra con la misma AWS KMS clave que se utilizó para cifrar la instantánea.
  - Si la instantánea indicada con `--snapshot-identifier` no está cifrada, el clúster restaurado no está cifrado.

Para 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
```

Para 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
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "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
    }
}
```

Una vez que el estado del clúster sea disponible, cree al menos una instancia para el clúster.

Para 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
```

Para 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
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "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"
    }
}
```

Restauración a un momento dado

Puede restaurar un clúster en cualquier momento que esté dentro del período de retención de copias de seguridad del clúster mediante la tecla Consola de administración de AWS o AWS Command Line Interface (AWS CLI).

> **Note**
>
> No puede restaurar a un momento específico un clúster 3.6 a uno 4.0, pero puede migrar de una versión de clúster a otra. Para obtener más información, consulte Migración y actualización de Amazon DocumentDB.

Tenga en cuenta las siguientes consideraciones al restaurar un clúster de base de datos a un momento dado.

- El nuevo clúster se crea con la misma configuración que el clúster de origen, con la salvedad de que el nuevo clúster se crea con el grupo de parámetros predeterminado. Para establecer el nuevo grupo de parámetros de clúster en el grupo de parámetros de clúster de origen, modifique el clúster en cuanto esté disponible. Para obtener más información acerca de la modificación de un clúster, consulte Modificación de un clúster de Amazon DocumentDB.

Using the Consola de administración de AWS

Para restaurar un clúster a un punto en el tiempo comprendido en el período de retención de copia de seguridad, siga este procedimiento en la Consola de administración de AWS.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Clusters (Clústeres). En la lista de clústeres, elija el botón situado a la izquierda del clúster que desee restaurar.

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En el menú Actions (Acciones), seleccione Restore to point in time (Restaurar a un momento dado).
4. Rellene la sección Restore time (Hora de restauración), en la que se especifican la fecha y la hora a la que restaurar.

   a. Fecha de restauración: elija o introduzca una fecha que esté entre la hora de restauración más temprana y la hora de restauración más reciente.

   b. Hora de restauración: elija o introduzca la hora, el minuto y los segundos que están entre la hora de restauración más temprana y la hora de restauración más reciente.
5. Rellene la sección Configuration (Configuración).

   a. Identificador de clúster: acepte el identificador predeterminado o introduzca otro identificador de su elección.

      Restricciones en cuanto a la nomenclatura de los clústeres:

      - Debe tener [1-63] letras, números o guiones.
      - El primer carácter debe ser una letra.
      - No puede terminar con un guion ni contener dos guiones consecutivos.
      - Debe ser único para todos los clústeres de Amazon RDS, Neptune y Amazon DocumentDB por región. Cuenta de AWS

   b. Clase de instancia: en la lista desplegable, elija la clase de instancia que desea utilizar para las instancias del clúster.

   c. Número de instancias: en la lista desplegable, elija el número de instancias que desee crear cuando se restaure el clúster.
6. En Configuración del almacenamiento en clúster, elija una opción de almacenamiento.

   > **Note**
   >
   > La configuración de I/O-Optimized almacenamiento de Amazon DocumentDB solo está disponible en las versiones 5.0 y 8.0 del motor de Amazon DocumentDB.
7. Opcional. Para configurar los ajustes de red y las opciones de clúster y para habilitar las exportaciones de registros, elija Show advanced settings (Mostrar configuración avanzada) y complete dichas secciones. De lo contrario, continúe en el siguiente paso.

   - Network settings (Configuración de red)

1. Nube privada virtual (VPC): en la lista desplegable, elija la VPC que quiera usar para este clúster.
2. Grupo de subredes: en la lista desplegable, elija el grupo de subredes para este clúster.
3. Grupos de seguridad de la VPC: en la lista desplegable, elija el grupo de seguridad de la VPC para este clúster.

- Cluster options (Opciones de clúster)
  1. Puerto: acepte el puerto predeterminado (27017) o utilice las flechas hacia arriba y abajo para establecer el puerto de comunicación con este clúster.

- Log exports (Exportaciones de registros)
  1. Registros de auditoría: seleccione esta opción para permitir la exportación de registros de auditoría a Amazon CloudWatch Logs. Si selecciona esta opción, debe habilitar `audit_logs` en el grupo personalizado de parámetros de clúster. Para obtener más información, consulte Auditoría de eventos de Amazon DocumentDB.
  2. Registros del generador de perfiles: seleccione esta opción para permitir la exportación de los registros del generador de perfiles de operaciones a Amazon CloudWatch Logs. Si selecciona esta opción, también debe modificar los siguientes parámetros del grupo personalizado de parámetros del clúster:
     - `profiler`: se establece en `enabled`.
     - `profiler_threshold_ms`: se establece en un valor `[0-INT_MAX]` para definir el umbral para las operaciones de creación de perfiles.
     - `profiler_sampling_rate`: se establece en un valor `[0.0-1.0]` para definir el porcentaje de operaciones lentas en el perfil.

     Para obtener más información, consulte Creación de perfiles de operaciones en Amazon DocumentDB.
  3. Registros del generador de perfiles: exporte los registros del generador de perfiles a Amazon CloudWatch
  4. Rol de IAM: en la lista desplegable, seleccione Rol vinculado al servicio RDS.
- Etiquetas

1. Agregar etiqueta: en el cuadro Clave, escriba el nombre de la etiqueta del clúster. En el cuadro Value (Valor), si lo desea, escriba el valor de la etiqueta. Las etiquetas se utilizan con las políticas de AWS Identity and Access Management (IAM) para administrar el acceso a los recursos de Amazon DocumentDB y para controlar qué acciones se pueden aplicar a los recursos.

- Protección contra eliminación

1. Habilitar la protección contra la eliminación: protege el clúster para que no se pueda eliminar accidentalmente. Cuando esta opción está habilitada, no se puede eliminar el clúster.

8. Para restaurar el clúster, elija Create cluster (Crear clúster). También puede elegir Cancel (Cancelar) para cancelar la operación.

Using the AWS CLI

Para restaurar un clúster a un momento dado mediante el periodo de retención de copia de seguridad de la instantánea, utilice la operación `restore-db-cluster-to-point-in-time` con los siguientes parámetros.

- **`--db-cluster-identifier`**: obligatorio. El nombre del nuevo clúster que se va a crear. Este clúster no puede existir antes de la operación. El valor del parámetro debe cumplir las siguientes restricciones.

  Restricciones en cuanto a la nomenclatura de los clústeres:

  - Debe tener [1-63] letras, números o guiones.
  - El primer carácter debe ser una letra.
  - No puede terminar con un guion ni contener dos guiones consecutivos.
  - Debe ser único para todos los clústeres de Amazon RDS, Neptune y Amazon DocumentDB por región. Cuenta de AWS

- **`--restore-to-time`**: la fecha y la hora UTC en que se va a restaurar el clúster. Por ejemplo, `2018-06-07T23:45:00Z`.

  Restricciones de tiempo:

  - Debe ser anterior a la última hora restaurable del clúster.
  - Debe especificarse si no se proporciona el parámetro `--use-latest-restorable-time`.
  - No se puede especificar si el parámetro `--use-latest-restorable-time` es `true`.

  - No se puede especificar si el valor del parámetro `--restore-type` es `copy-on-write`.
- **`--source-db-cluster-identifier`**: el nombre del clúster de origen desde el que se va a restaurar. Este clúster debe existir y estar disponible.
- **`--use-latest-restorable-time`** o **`--no-use-latest-restorable-time`**: si se va a restaurar a la hora más reciente de la copia de seguridad restaurable. No se puede especificar si se proporciona el parámetro `--restore-to-time`.
- **`--storage-type standard | iopt1`**: opcional. Predeterminado: `standard`.

La AWS CLI operación `restore-db-cluster-to-point-in-time` solo restaura el clúster, no las instancias de ese clúster. Debe invocar la operación `create-db-instance` para crear instancias para el clúster restaurado, especificando el identificador de dicho clúster restaurado en `--db-cluster-identifier`. Solo puede crear instancias después de que se haya completado la operación `restore-db-cluster-to-point-in-time` y de que el clúster restaurado esté disponible.

Example

En el siguiente ejemplo, `sample-cluster-restored` se crea a partir de la instantánea `sample-cluster-snapshot` tomando el último momento que se puede restaurar.

Para 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
```

Para 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

En el siguiente ejemplo, `sample-cluster-restored` se crea a partir de la instantánea `sample-cluster-snapshot` en el punto en el que estaba a las 03:15 del 11 de diciembre

de 2018 (UTC), que está dentro del periodo de retención de copia de seguridad de `sample-cluster`.

Para 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
```

Para 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
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "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
    }
}
```

Eliminación de una instantánea del clúster

Una instantánea manual es una copia de seguridad completa que solo se elimina cuando se elimina manualmente mediante el comando Consola de administración de AWS o AWS CLI. No se puede eliminar manualmente una instantánea automática, porque las instantáneas automáticas solo se eliminan cuando finaliza el período de retención o cuando elimina el clúster de la instantánea.

Using the Consola de administración de AWS

Para eliminar una instantánea de clúster manual mediante el Consola de administración de AWS, siga estos pasos.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, elija Snapshots (Instantáneas).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En la lista de instantáneas, elija el botón situado a la izquierda de la instantánea que desea eliminar. El tipo de la instantánea debe ser manual.

    1. Puede verificar que el tipo de instantánea es manual comprobando si aparece enumerado como `manual` o `automatic` en la columna Tipo.
4. En el menú Actions (Acciones), elija Delete (Eliminar). Si la opción Delete (Eliminar) no está disponible, lo más probable es que haya elegido una instantánea automática.
5. En la pantalla de confirmación de eliminación, para eliminar la instantánea, seleccione Delete (Eliminar). Para conservar la instantánea, elija Cancel (Cancelar).

Using the AWS CLI

Una instantánea manual del clúster de Amazon DocumentDB es una copia de seguridad completa que puede eliminar manualmente mediante la AWS CLI. No puede eliminar manualmente una instantánea automática.

Para eliminar una instantánea de clúster manual mediante AWS CLI, utilice la `delete-db-cluster-snapshot` operación con los siguientes parámetros.

Parameters

- **`--db-cluster-snapshot-identifier`**: obligatorio. El nombre de la instantánea manual que se va a eliminar.

En el siguiente ejemplo se elimina la instantánea `sample-cluster-snapshot` del clúster:

Para Linux, macOS o Unix:

```
aws docdb delete-db-cluster-snapshot \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Para Windows:

```
aws docdb delete-db-cluster-snapshot ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

En la salida de esta operación, se muestran los detalles de la instantánea de clúster que ha eliminado.

Supervisión y registro en Amazon DocumentDB

La supervisión de sus AWS servicios es una parte importante para mantener sus sistemas en buen estado y funcionando de manera óptima. Recopile datos de supervisión de todas las partes de su AWS solución para poder depurar y corregir errores o problemas de rendimiento con mayor facilidad. Antes de empezar a supervisar AWS las soluciones, plantéese las siguientes preguntas:

- ¿Cuáles son los objetivos de la supervisión?
- ¿Qué recursos va a monitorizar?
- ¿Con qué frecuencia va a monitorizar estos recursos?
- ¿Qué herramientas de supervisión va a utilizar?
- ¿Quién es el responsable de realizar el seguimiento?
- ¿A quién se va a notificar y qué ocurre si algo va mal?

Para comprender los patrones de rendimiento actual, identificar las anomalías de rendimiento y elaborar métodos para la resolución de problemas, debe establecer métricas de rendimiento de referencia para distintos momentos y bajo diferentes condiciones de carga. Mientras monitoriza su AWS solución, almacene sus datos de monitoreo históricos para consultarlos en el futuro y establecer sus puntos de referencia.

En general, los valores aceptables para las métricas de desempeño dependen del aspecto de la referencia y de lo que hace la aplicación. Investigue las variaciones coherentes o de las tendencias con respecto a la referencia. A continuación, se ofrecen algunas sugerencias sobre tipos concretos de métricas:

- Consumo elevado de CPU o RAM: unos valores elevados de consumo de CPU o RAM pueden ser adecuados si se ajustan a los objetivos de su aplicación (de rendimiento o simultaneidad, por ejemplo) y son los esperados.
- Consumo de volumen de almacenamiento: investigue el consumo de almacenamiento (`VolumeBytesUsed`) si el espacio utilizado está por sistema alrededor o por encima del 85 % del espacio del volumen de almacenamiento. Determine si es posible eliminar datos del volumen de almacenamiento o archivar los datos en un sistema diferente para liberar espacio. Para obtener más información, consulte Almacenamiento de Amazon DocumentDB y Cuotas de Amazon DocumentDB.

- Tráfico de red: para el tráfico de red, hable con el administrador de su sistema para saber cuál es el rendimiento esperado para la red de su dominio y para su conexión a Internet. Investigue el tráfico de red si el rendimiento es por sistema inferior al esperado.
- Conexiones a bases de datos: valore la posibilidad de restringir las conexiones a las bases de datos si ve que hay un alto número de conexiones de usuarios junto con una reducción en el rendimiento y el tiempo de respuesta de la instancia. El mejor número de conexiones de usuarios para su instancia variará en función de la clase de instancia y de la complejidad de las operaciones que se estén llevando a cabo.
- Métricas de IOPS: los valores esperados para las métricas de IOPS dependen de la especificación del disco y la configuración del servidor, así que debe usar su referencia para conocer los valores típicos. Investigue si los valores son por sistema diferentes de los valores de referencia. Para un rendimiento óptimo de IOPS, asegúrese de que el conjunto de trabajo típico se ajuste a la memoria para minimizar las operaciones de lectura y escritura.

Amazon DocumentDB (compatible con MongoDB) proporciona una variedad de CloudWatch métricas de Amazon que puede supervisar para determinar el estado y el rendimiento de sus clústeres e instancias de Amazon DocumentDB. Puede ver las métricas de Amazon DocumentDB con varias herramientas, como la consola de Amazon DocumentDB AWS CLI, CloudWatch la API y Performance Insights.

Temas

- Supervisión del estado de un clúster de Amazon DocumentDB
- Supervisión del estado de un clúster de Amazon DocumentDB
- Visualización de recomendaciones de Amazon DocumentDB
- Uso de suscripciones a eventos de Amazon DocumentDB
- Supervisión de Amazon DocumentDB con CloudWatch
- Registro de llamadas a la API de Amazon DocumentDB con AWS CloudTrail
- Creación de perfiles de operaciones en Amazon DocumentDB
- Supervisión con información sobre rendimiento

Supervisión del estado de un clúster de Amazon DocumentDB

El estado de un clúster indica la situación de este. Puede ver el estado de un clúster mediante la consola de Amazon DocumentDB o el comando. AWS CLI `describe-db-clusters`

Temas

- Valores de estado del clúster
- Monitorización del estado de un clúster

Valores de estado del clúster

En la siguiente tabla se muestran los valores válidos para el estado de un clúster.

Estado del clúster	Description (Descripción)
`active`	El clúster está activo. Este estado se aplica únicamente a los clústeres elásticos.
`available`	El clúster funciona correctam ente y está disponible. Este estado solo se aplica a clústeres basados en instancia s.
`backing-up`	Se está creando una copia de seguridad del clúster.
`creating`	El clúster se está creando. No se puede obtener acceso a él mientras se está creando.
`deleting`	El clúster se está eliminando. No se puede obtener acceso a él mientras se está eliminando.
`failing-over`	Se está realizando una conmutación por error de la instancia principal a una réplica de Amazon DocumentDB.

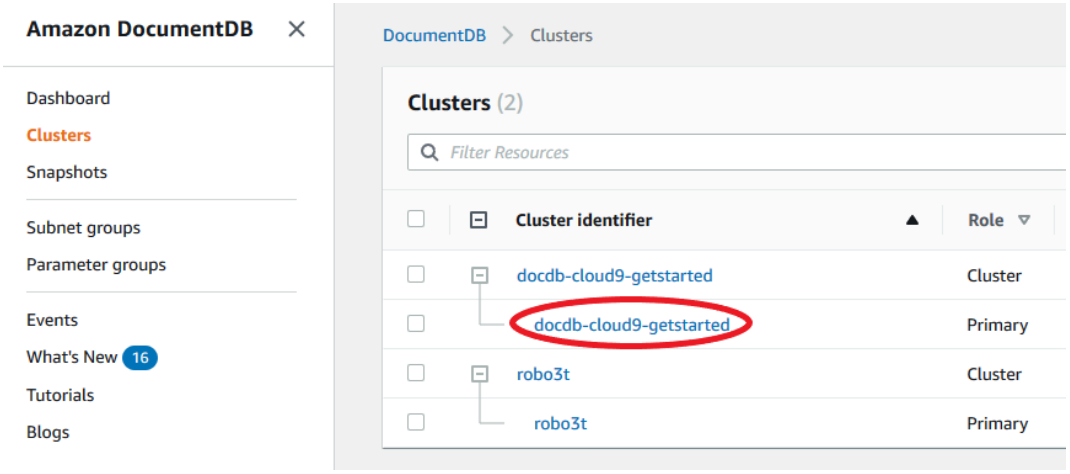
Estado del clúster	Description (Descripción)
`inaccessible-encryption-credentials`	No se puede acceder a la AWS KMS clave utilizada para cifrar o descifrar el clúster.
`maintenance`	Se está aplicando una actualización de mantenimi ento al clúster. Este estado se usa para el mantenimiento de nivel de clúster que Amazon DocumentDB programa con mucha antelación.
`migrating`	Se está restaurando una instantánea de clúster en un clúster.
`migration-failed`	Una migración no se ha realizado correctamente.
`modifying`	El clúster se está modificando porque un cliente ha solicitado su modificación.
`renaming`	El nombre del clúster se está cambiando porque un cliente lo ha solicitado.
`resetting-master-credentials`	Las credenciales maestras del clúster se están restableciendo porque un cliente lo ha solicitad o.
`upgrading`	Se está actualizando la versión del motor del clúster.

Monitorización del estado de un clúster

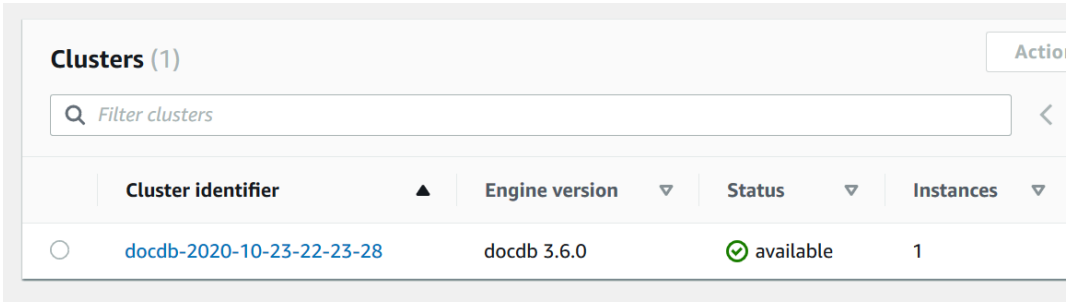
Using the Consola de administración de AWS

Cuando utilice el Consola de administración de AWS para determinar el estado de un clúster, utilice el procedimiento siguiente.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Clusters (Clústeres).
3. En el cuadro de navegación de clústeres, verá la columna Identificador del clúster. Sus instancias se enumeran en clústeres, de forma similar a la siguiente captura de pantalla.



4. En la columna Identificador de clúster, busque el nombre de la instancia que le interese. A continuación, para encontrar el estado de la instancia, consulte en esa fila la columna Estado, tal y como se muestra a continuación.



Using the AWS CLI

Cuando utilices el AWS CLI para determinar el estado de un clúster, utiliza la `describe-db-clusters` operación. El siguiente código busca el estado del clúster `sample-cluster`.

Para Linux, macOS o Unix:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster  \
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Para Windows:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster  ^
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

La salida de esta operación será similar a lo que se indica a continuación.

```
[
    [
        "sample-cluster",
        "available"
    ]
]
```

Supervisión del estado de un clúster de Amazon DocumentDB

Amazon DocumentDB proporciona información sobre el estado actual de cada instancia configurada de la base de datos.

Hay tres tipos de estado que puede ver para una instancia de Amazon DocumentDB:

- Estado de la instancia: este estado se muestra en la columna Estado de la tabla de clústeres Consola de administración de AWS y muestra el estado actual del ciclo de vida de la instancia. Los valores que se muestran en la columna Estado se derivan del campo `Status` de la respuesta de la API de `DescribeDBCluster`.
- Estado de la instancia: este estado se muestra en la columna de estado de la instancia de la tabla de clústeres en Consola de administración de AWS y muestra si el motor de base de datos, el componente responsable de administrar y recuperar los datos, está en ejecución. Los valores que se muestran en la columna de estado de la instancia se basan en la métrica del CloudWatch `EngineUptime` sistema de Amazon.

- Estado de mantenimiento: este estado se muestra en la columna Mantenimiento de la tabla de clústeres Consola de administración de AWS e indica el estado de cualquier evento de mantenimiento que deba aplicarse a una instancia. El estado de mantenimiento es independiente del estado de las demás instancias y se deriva de la API de `PendingMaintenanceAction`. Para obtener más información sobre el estado de mantenimiento, consulte Mantenimiento de Amazon DocumentDB.

Temas

- Valores de estado de instancia
- Supervise el estado de la instancia mediante el Consola de administración de AWS o AWS CLI
- Estado de una instancia
- Supervisar el estado de la instancia mediante el Consola de administración de AWS

Valores de estado de instancia

En la siguiente tabla se muestran los posibles valores de estado de las instancias y cómo se factura para cada estado. Muestra si se le facturará la instancia y el almacenamiento, solo el almacenamiento o si no se le facturará. Para todos los estados de instancia, se le factura siempre el uso de copia de seguridad.

estado de instancias	Facturado	Description (Descripción)
`available`	Facturado	La instancia funciona correctamente y está disponible.
`backing-up`	Facturado	Se está creando una copia de seguridad de la instancia.
`configuring-log-exports`	Facturado	La publicación de archivos de registro en Amazon CloudWatch Logs está habilitada o deshabilitada para esta instancia.
`creating`	No facturado	La instancia se está creando. No se puede obtener acceso a la instancia mientras se está creando.

estado de instancias	Facturado	Description (Descripción)
`deleting`	No facturado	La instancia se está eliminando.
`failed`	No facturado	La instancia ha generado un error y Amazon DocumentDB no ha podido recuperar la. Para recuperar los datos, realice una restauración al último momento restaurable de la instancia.
`inaccessible-encryption-credentials`	No facturado	No se pudo acceder a la AWS KMS clave que se usa para cifrar o descifrar la instancia.
`incompatible-network`	No facturado	Amazon DocumentDB está intentando realizar una acción de recuperación en una instancia, pero no puede hacerlo porque la VPC está en un estado que impide completar la acción. Este estado puede darse si, por ejemplo, todas las direccion es IP disponibles en una subred estaban en uso y Amazon DocumentDB no puede obtener una dirección IP para la instancia.
`maintenance`	Facturado	Amazon DocumentDB está aplicando una actualización de mantenimiento a la instancia. Este estado se usa para el mantenimiento de nivel de instancia que Amazon DocumentDB programa con mucha antelación. Estamos evaluando formas de exponer otras acciones de mantenimiento a los clientes a través de este estado.
`modifying`	Facturado	La instancia se está modificando debido a una solicitud para modificar la instancia.

estado de instancias	Facturado	Description (Descripción)
`rebooting`	Facturado	La instancia se está reiniciando debido a una solicitud o un proceso de Amazon DocumentDB que requiere el reinicio de la instancia.
`renaming`	Facturado	El nombre de la instancia se está cambiando debido a una solicitud de cambio de nombre.
`resetting-master-credentials`	Facturado	Las credenciales maestras de la instancia se están restableciendo debido a una solicitud de restablecimiento.
`restore-error`	Facturado	La instancia ha registrado un error al intentar restaurar a un momento dado o a partir de una instantánea.
`starting`	Facturado para almacenamiento	La instancia se está iniciando.
`stopped`	Facturado para almacenamiento	La instancia se ha detenido.
`stopping`	Facturado para almacenamiento	La instancia se está deteniendo.
`storage-full`	Facturado	La instancia ha alcanzado su asignación de capacidad de almacenamiento. Es un estado crítico y se debe corregir de inmediato. Escale verticalmente el almacenamiento modificando la instancia. Configura CloudWatch las alarmas de Amazon para que te avisen cuando el espacio de almacenamiento se esté agotando para que no te encuentres en esta situación.

Supervise el estado de la instancia mediante el Consola de administración de AWS o AWS CLI

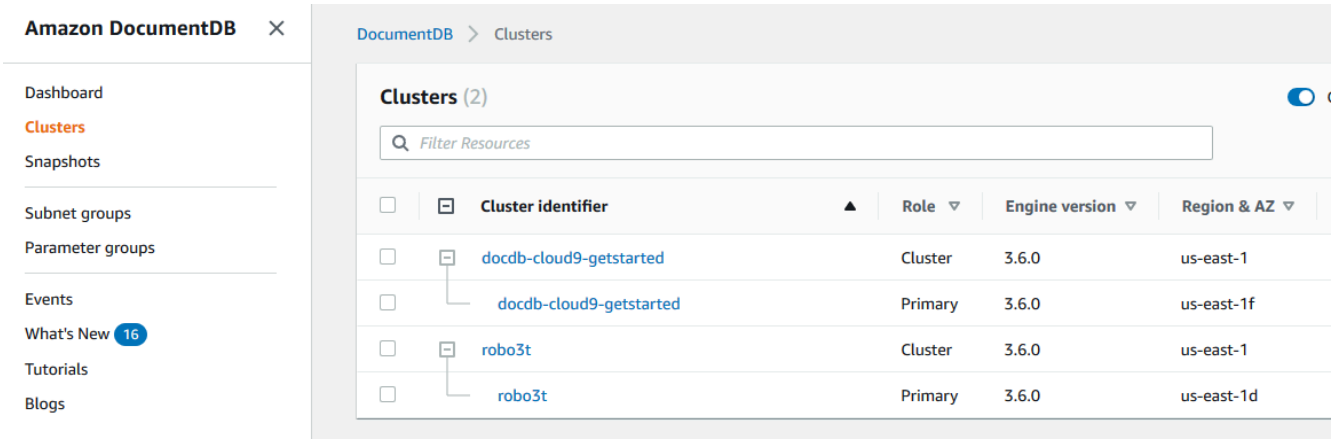
Usa el Consola de administración de AWS o AWS CLI para supervisar el estado de tu instancia.

Using the Consola de administración de AWS

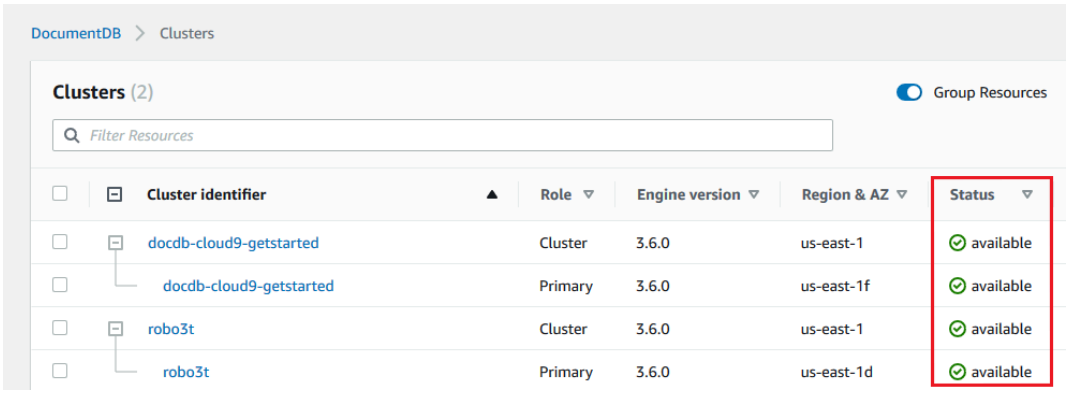
Cuando la utilice Consola de administración de AWS para determinar el estado de un clúster, utilice el procedimiento siguiente.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Clusters (Clústeres).

> **Note**
>
> Tenga en cuenta que en el cuadro de navegación de clústeres, la columna Identificador de clústeres muestra tanto los clústeres como las instancias. Las instancias aparecen debajo de los clústeres, de forma similar a la imagen siguiente.



3. Busque el nombre de la instancia que le interesa. A continuación, para encontrar el estado de la instancia, consulte en esa fila la columna Status (Estado), tal y como se muestra a continuación.



Using the AWS CLI

Cuando utilice el AWS CLI para determinar el estado de un clúster, utilice la `describe-db-instances` operación. El siguiente código muestra el estado de la instancia `sample-cluster-instance-01`.

Para Linux, macOS o Unix:

```
aws docdb describe-db-instances \
          --db-instance-identifier sample-cluster-instance-01  \
          --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]'
```

Para Windows:

```
aws docdb describe-db-instances ^
          --db-instance-identifier sample-cluster-instance-01  ^
          --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]'
```

La salida de esta operación será similar a lo que se indica a continuación.

```
[
          [
              "sample-cluster-instance-01",
              "available"
          ]
      ]
```

Estado de una instancia

Encuentre los valores de estado posibles para instancias de base de datos en la siguiente tabla. La columna Estado de la instancia, que se encuentra en la tabla Clústeres del Consola de administración de AWS, muestra si el motor de base de datos, el componente responsable de almacenar, administrar y recuperar los datos, funciona con normalidad. Esta columna también indica si la métrica `EngineUptime` del sistema, disponible en CloudWatch, muestra el estado de cada instancia.

Estado de una instancia	Description (Descripción)
buen estado	El motor de base de datos se ejecuta en la instancia de Amazon DocumentDB.
Mal estado	El motor de base de datos no se está ejecutando o se reinició hace menos de un minuto.

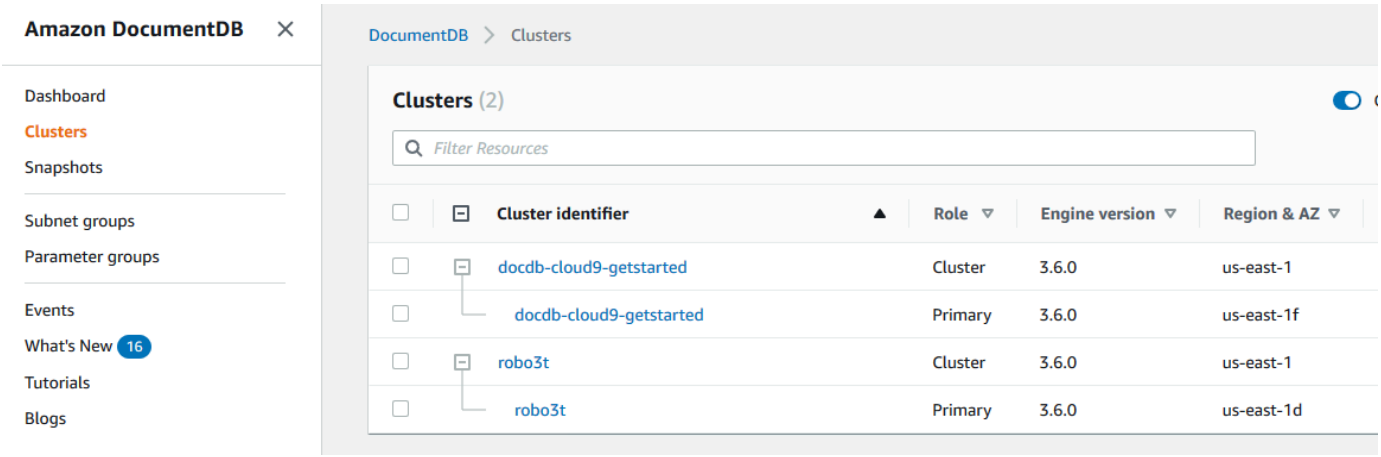
Supervisar el estado de la instancia mediante el Consola de administración de AWS

Consola de administración de AWS Utilízala para supervisar el estado de salud de tu instancia.

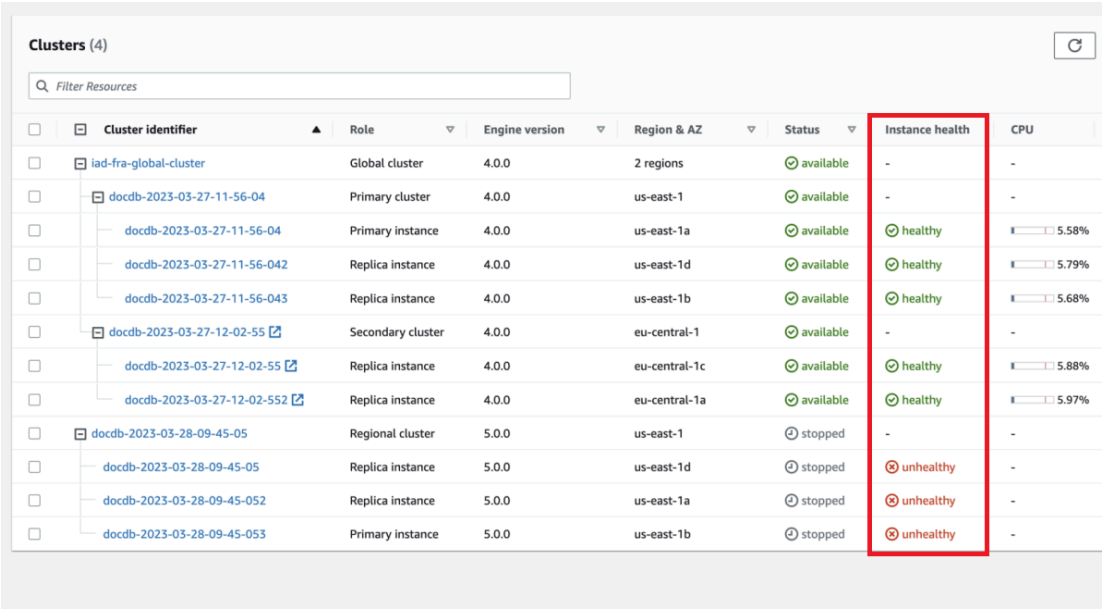
Cuando la utilices Consola de administración de AWS, sigue los pasos siguientes para conocer el estado de la instancia.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Clusters (Clústeres).

> **Note**
>
> Tenga en cuenta que en el cuadro de navegación Clústeres, la columna Identificador del clúster muestra tanto los clústeres como las instancias. Las instancias aparecen debajo de los clústeres, de forma similar a la imagen siguiente.



3. Busque el nombre de la instancia que le interesa. A continuación, para encontrar el estado de la instancia, consulte en esa fila la columna Estado, tal y como se muestra a continuación:



> **Note**
>
> El sondeo del estado de las instancias se realiza cada 60 segundos y se basa en la métrica CloudWatch `EngineUptime` del sistema. Los valores de la columna Estado de la instancia se actualizan automáticamente.

Visualización de recomendaciones de Amazon DocumentDB

Amazon DocumentDB proporciona una lista de recomendaciones automatizadas para recursos de base de datos, como instancias y clústeres. Estas recomendaciones proporcionan instrucciones de las prácticas recomendadas analizando los datos de rendimiento, el uso y la configuración de la instancia y clúster.

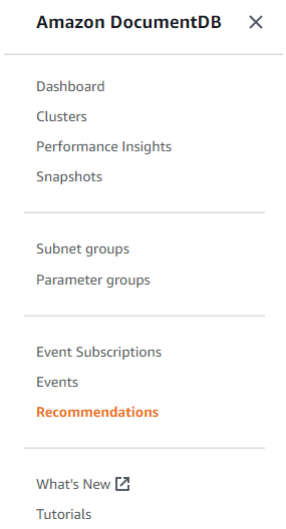
Para ver ejemplos de estas recomendaciones, consulte lo siguiente:

Tipo	Descripción	Recomendación	Información adicional
Una instancia	El clúster solo contiene una instancia	Rendimiento y disponibilidad: añada otra instancia con la misma clase de instancia en una zona de disponibilidad diferente.	Alta disponibilidad y replicación de Amazon DocumentDB

Amazon DocumentDB genera recomendaciones para un recurso cuando se crea o modifica el recurso. Amazon DocumentDB analiza también periódicamente sus recursos y genera recomendaciones.

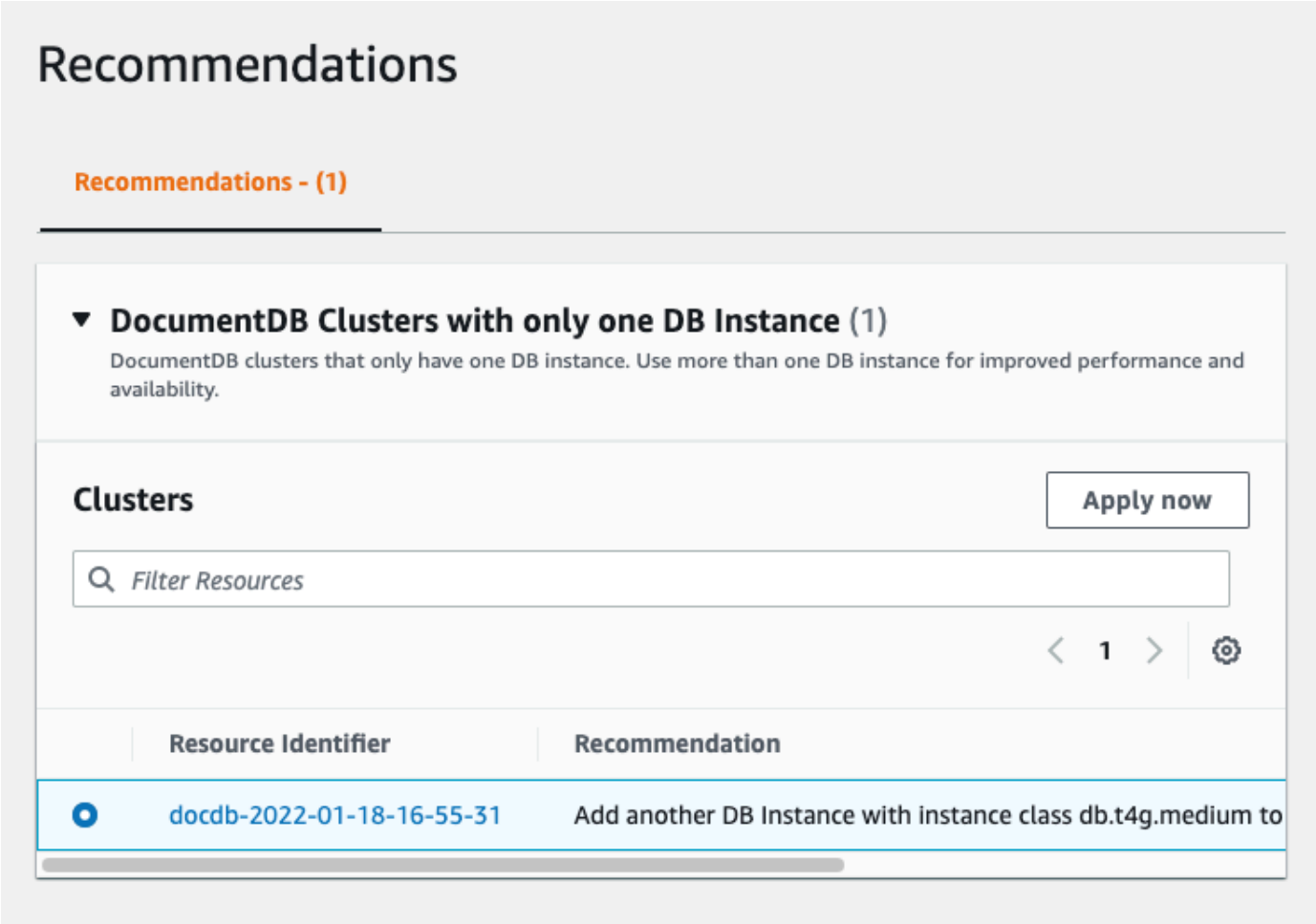
Visualización de las recomendaciones de Amazon DocumentDB y toma de medidas al respecto

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Recomendaciones:



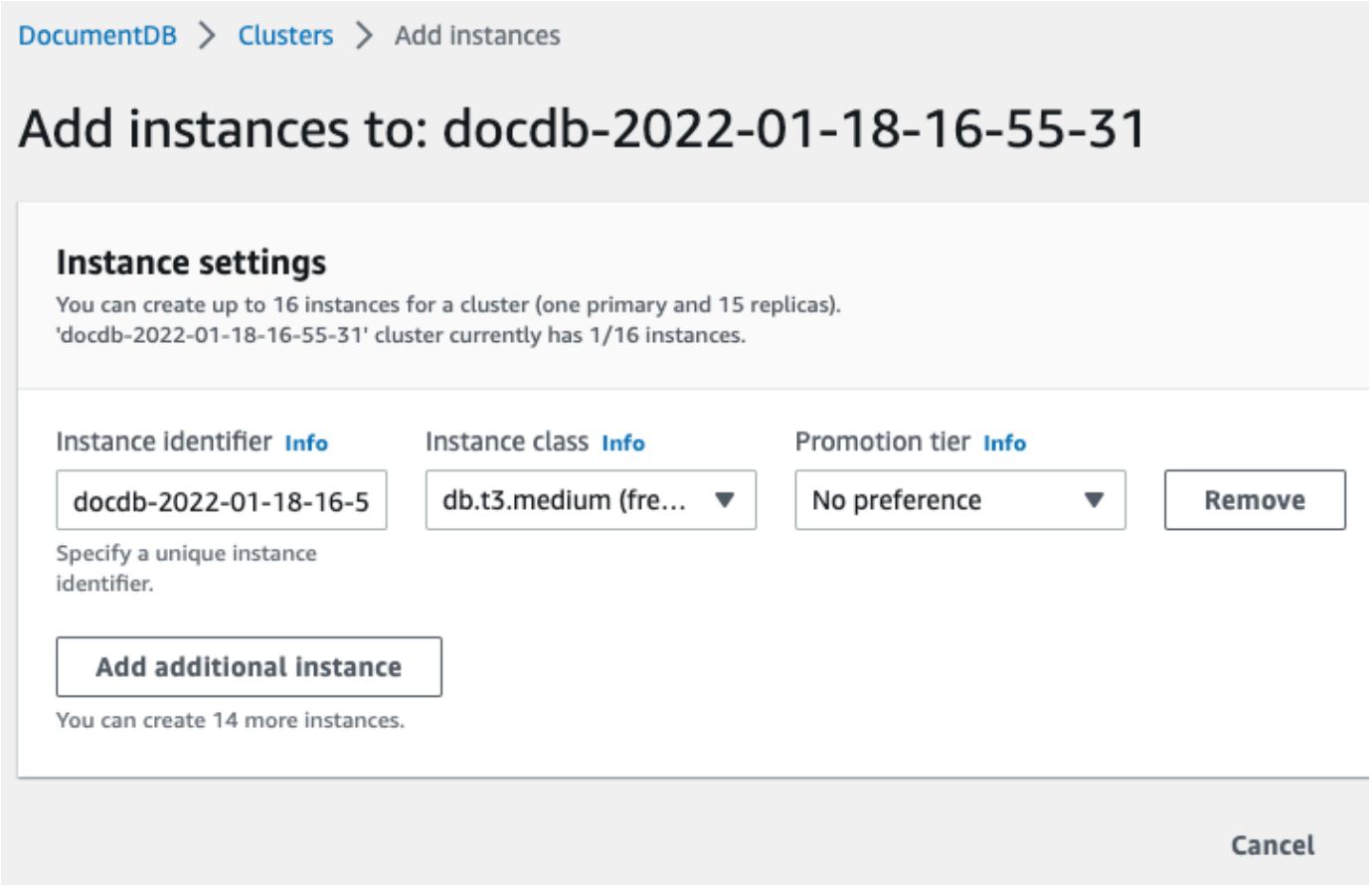
3. En el cuadro de diálogo Recomendaciones, amplíe la sección de interés y seleccione la tarea recomendada.

   En el siguiente ejemplo, la tarea recomendada se aplica a un clúster de Amazon DocumentDB con una sola instancia. La recomendación es añadir otra instancia para mejorar el rendimiento y la disponibilidad.



4. Elija Apply now.

   En este ejemplo, aparece el cuadro de diálogo Agregar instancias:



5. Modifique la configuración de la nueva instancia y elija Crear.

Uso de suscripciones a eventos de Amazon DocumentDB

Amazon RDS utiliza Amazon Simple Notification Service (Amazon SNS) para proporcionar notificaciones cuando se produce un evento de Amazon DocumentDB. Estas notificaciones pueden estar en cualquier formato compatible con Amazon SNS Región de AWS, como un correo electrónico, un mensaje de texto o una llamada a un punto de conexión HTTP.

Amazon DocumentDB agrupa estos eventos en categorías a las que puede suscribirse para recibir una notificación cada vez que se produzca un evento en esa categoría. Puede suscribirse a una categoría de eventos para una instancia, un clúster, una instantánea, una instantánea de clúster, o para un grupo de parámetros. Por ejemplo, si se suscribe a la categoría Backup de una instancia determinada, recibe una notificación cada vez que se produzca un evento relacionado con las copias de seguridad que afecte a dicha instancia. También recibirá una notificación cuando cambie una suscripción de eventos.

Los eventos se producen en el clúster y en el nivel de instancia, por lo que puede recibir eventos si se suscribe a un clúster o a una instancia.

Las suscripciones de eventos se envían a las direcciones que se proporcionan al crear la suscripción. Es posible que le interese crear distintas suscripciones como, por ejemplo, una que reciba todas las notificaciones de eventos y otra que incluya únicamente los eventos críticos para las instancias de producción. Puede desactivar fácilmente las notificaciones sin eliminar una suscripción. Para ello, defina el botón de opción Activado en No en la consola de Amazon DocumentDB.

> **Important**
>
> Amazon DocumentDB no garantiza el orden de los eventos enviados en una secuencia de eventos. El orden de los eventos está sujeto a cambio.

Amazon DocumentDB utiliza el nombre de recurso de Amazon (ARN) de un tema de Amazon SNS para identificar cada suscripción. La consola de Amazon DocumentDB crea el ARN automáticamente cuando se crea la suscripción.

La facturación de las suscripciones a eventos de Amazon DocumentDB se realiza a través de Amazon SNS. Se aplican las tarifas de Amazon SNS cuando se utiliza la notificación de eventos. Para obtener más información, consulte Amazon Simple Notification Service Pricing. Además de los cargos de Amazon SNS, Amazon DocumentDB no factura las suscripciones a eventos.

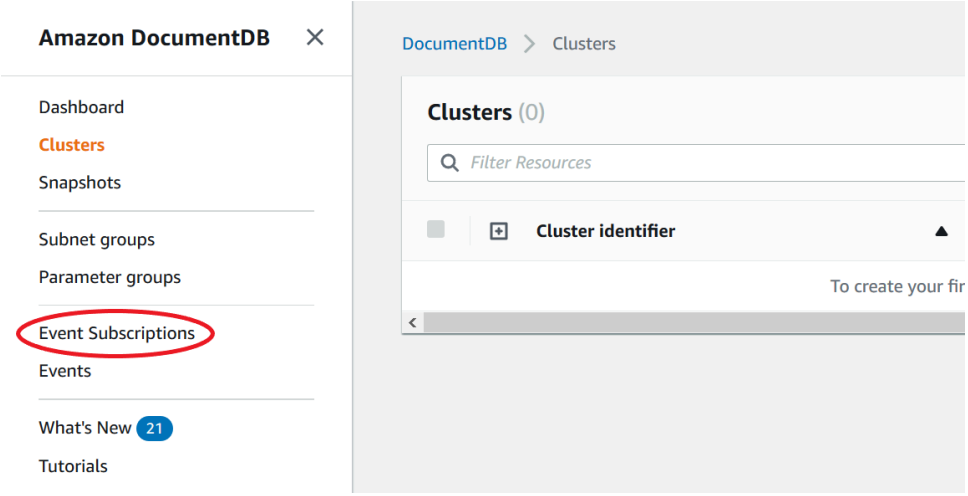
Temas

- Suscripción a eventos de Amazon DocumentDB
- Administración de suscripciones a notificaciones de eventos de Amazon DocumentDB
- Categorías y mensajes de eventos de Amazon DocumentDB

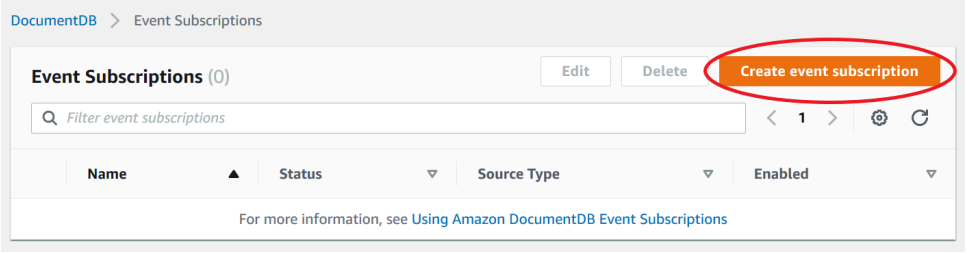
Suscripción a eventos de Amazon DocumentDB

Puede utilizar la consola Amazon DocumentDB para suscribirse a las suscripciones a eventos de la siguiente manera:

1. Inicie sesión Consola de administración de AWS enhttps://console.aws.amazon.com/docdb.
2. En el panel de navegación seleccione Event Subscriptions (Suscripciones de eventos).

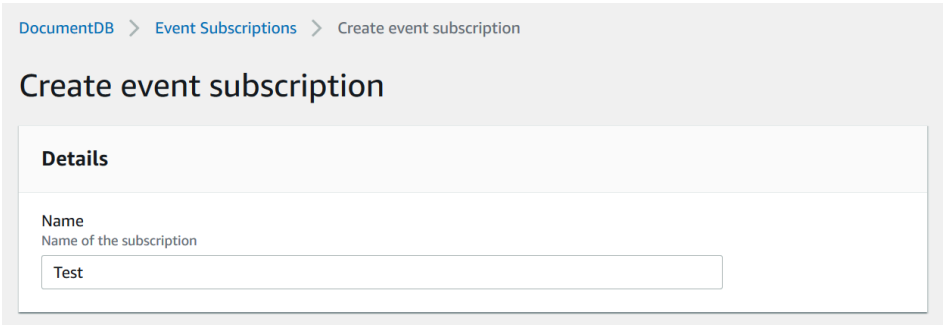


3. En la página Event Subscriptions (Suscripciones de eventos) seleccione Create Event Subscription (Crear suscripción de eventos).

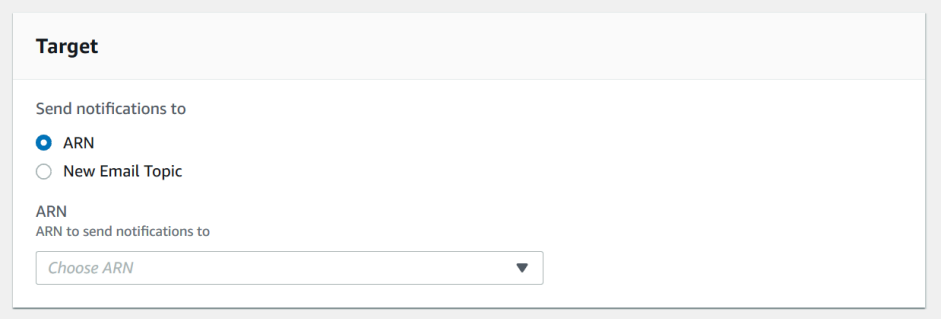


4. En el cuadro de diálogo Create Event Subscription (Crear suscripción de eventos), haga lo siguiente:

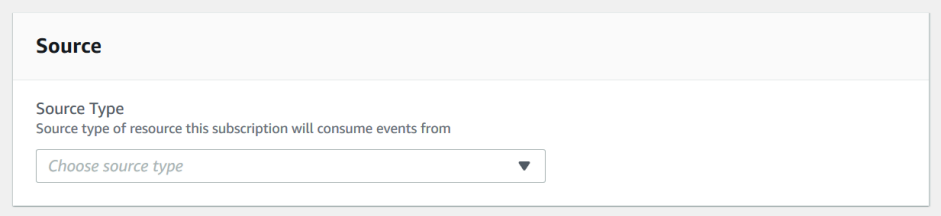
   - En Name (Nombre) escriba un nombre para la suscripción de notificación de evento.



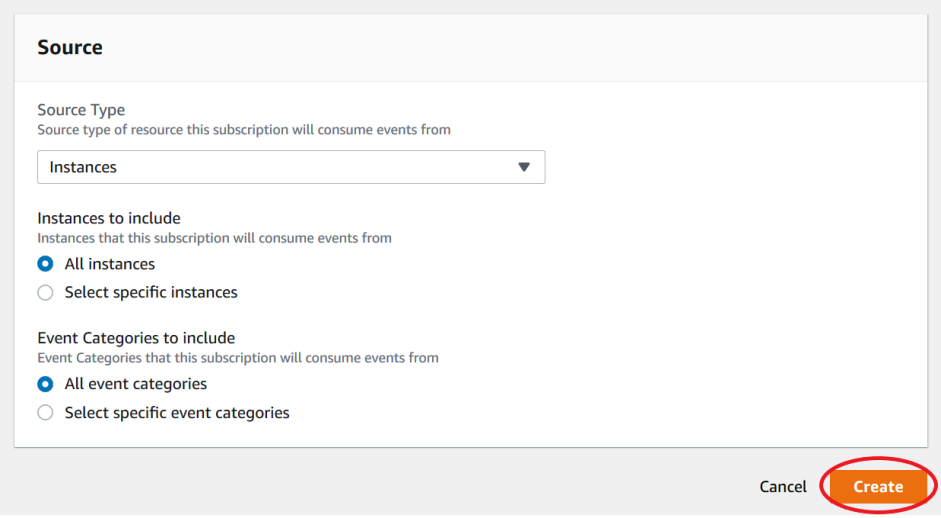
   - En Destino, seleccione a dónde quiere enviar las notificaciones. Puede elegir un ARN existente o elegir Nuevo tema de correo electrónico para introducir el nombre de un tema y una lista de destinatarios.



- En Origen elija un tipo de origen. En función del tipo de origen que haya seleccionado, seleccione las categorías y orígenes del evento de las que desea recibir notificaciones.



- Seleccione Crear.

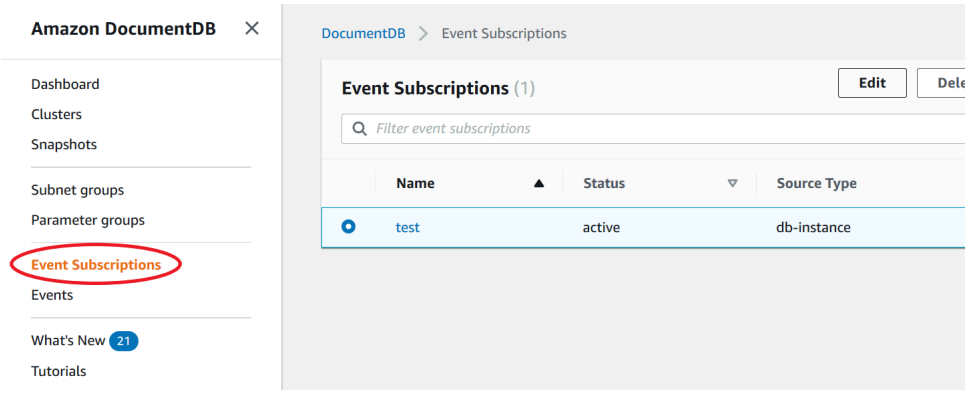


Administración de suscripciones a notificaciones de eventos de Amazon DocumentDB

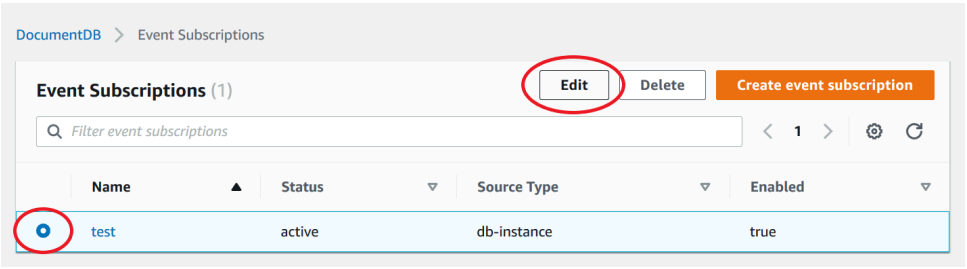
Si selecciona Suscripciones a eventos en el panel de navegación de la consola de Amazon DocumentDB, puede ver las categorías de suscripciones y una lista de sus suscripciones actuales. También puede modificar o eliminar una suscripción específica.

Modificación de sus suscripciones actuales de notificación de eventos de Amazon DocumentDB

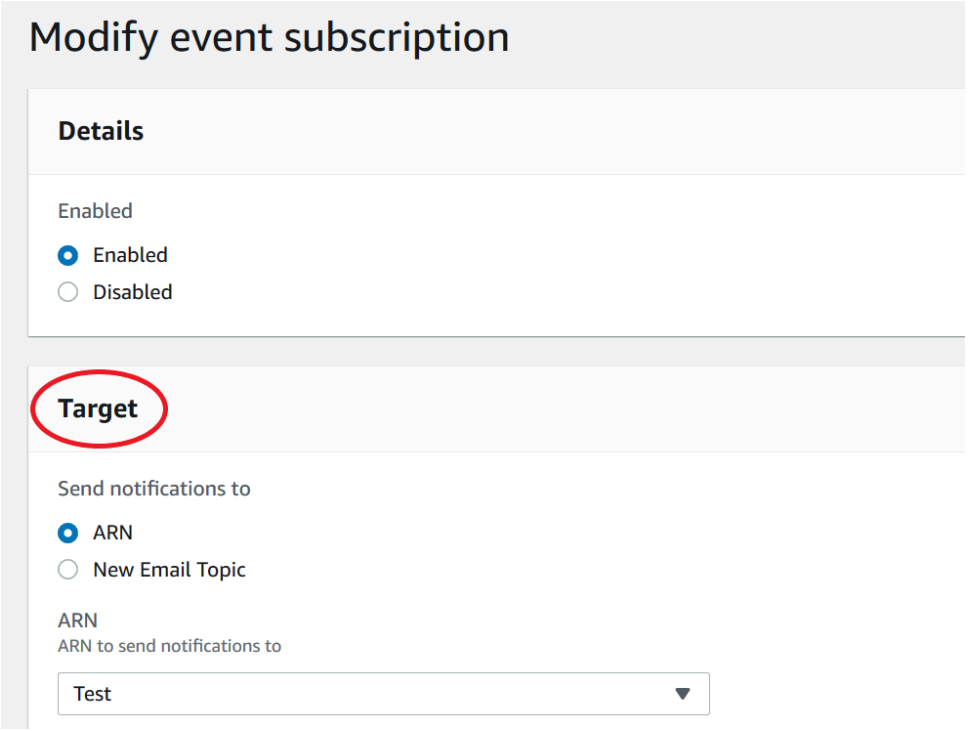
1. Inicie sesión en el correo Consola de administración de AWS electrónicohttps://console.aws.amazon.com/docdb.
2. En el panel de navegación seleccione Event Subscriptions (Suscripciones de eventos). El panel Event subscriptions (Suscripciones de eventos) muestra todas sus suscripciones a notificaciones de eventos.



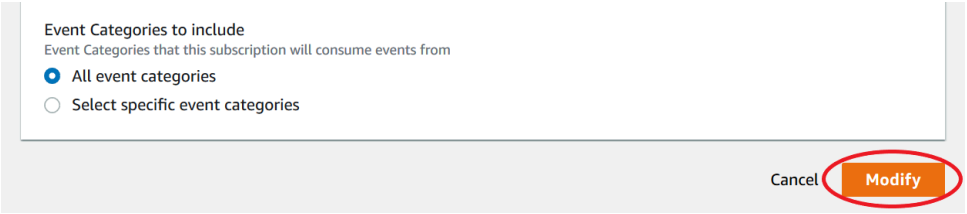
3. En el panel Event subscriptions (Suscripciones de eventos), elija la suscripción que desea modificar y elija Edit (Editar).



4. Realice los cambios que desee en la suscripción en las secciones Target (Objetivo) o Source (Fuente). Puede añadir o eliminar identificadores de origen activándolos o desactivándolos en la sección Origen.



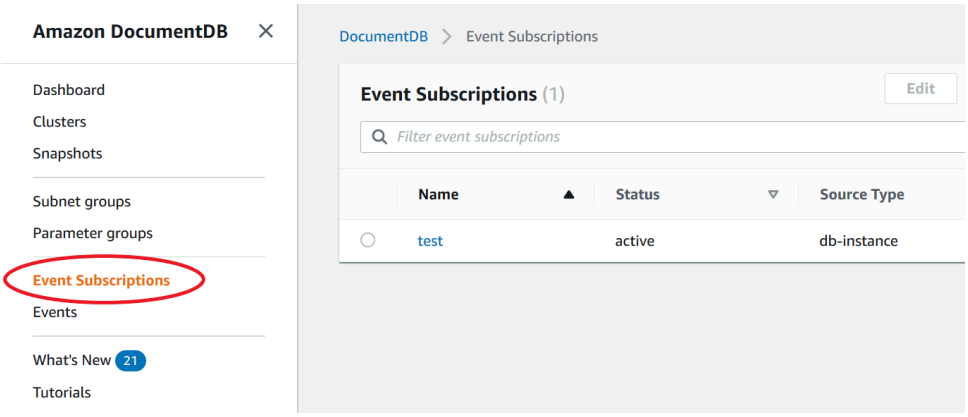
5. Elija Modificar. La consola de Amazon DocumentDB indica que se está modificando la suscripción.



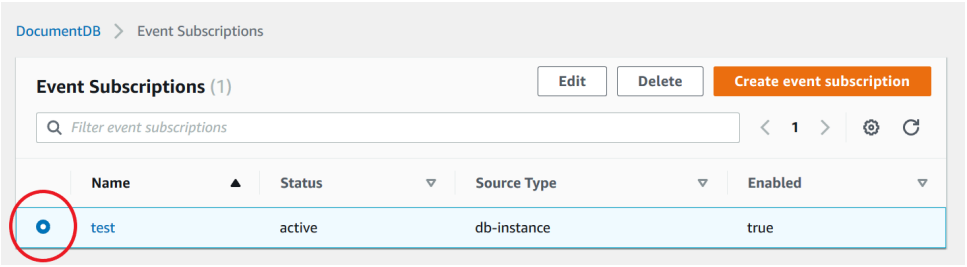
Eliminación de una suscripción de notificación de eventos de Amazon DocumentDB

Puede eliminar una suscripción cuando ya no la necesite. Los suscriptores del tema dejarán de recibir notificaciones de los eventos especificados en la suscripción.

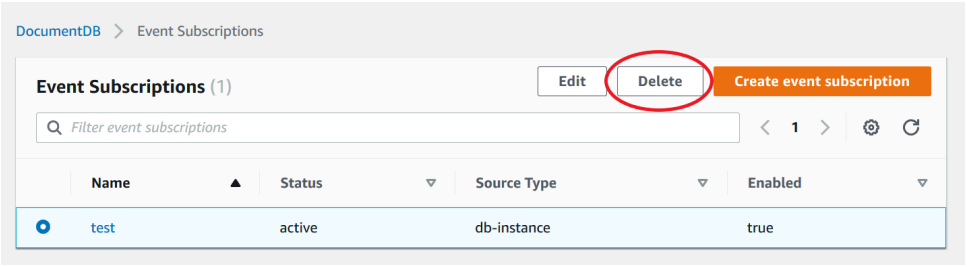
1. Inicie sesión en el correo Consola de administración de AWS electrónicohttps://console.aws.amazon.com/docdb.
2. En el panel de navegación seleccione Event Subscriptions (Suscripciones de eventos).



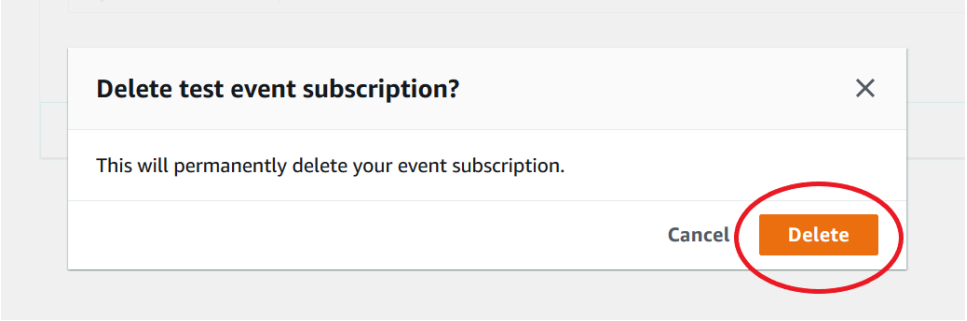
3. En el panel Suscripciones a eventos, seleccione la suscripción que desea eliminar.



4. Elija Eliminar.



5. Aparecerá una ventana emergente en la que se le preguntará si desea eliminar esta notificación de forma permanente. Elija Eliminar.



Categorías y mensajes de eventos de Amazon DocumentDB

Amazon DocumentDB genera un número significativo de eventos en categorías a las que puede suscribirse a través de la consola. Cada categoría se aplica a un tipo de origen, que puede ser una instancia, un clúster, una instantánea o un grupo de parámetros.

> **Note**
>
> Amazon DocumentDB utiliza las definiciones e ID de eventos de Amazon RDS existentes.

Eventos de Amazon DocumentDB que se originan en instancias

Categoría	Description (Descripción)
availability	Instancia reiniciada.
disponibilidad	Instancia cerrada.
cambio de configuración	Se está aplicando la modificación a la clase de instancia.
cambio de configuración	Ha finalizado la aplicación de la modificación a una clase de instancia.
cambio de configuración	Restablezca las credenciales principales.
creación	Instancia creada.
eliminación	Se ha eliminado la instancia
error	Se ha producido un error en la instancia debido a una configuración incompatible o a un problema de almacenamiento subyacent e. Comience una restauración a un momento dado para la instancia.
notificación	Se ha detenido la instancia.
notificación	Se ha iniciado la instancia.

Categoría	Description (Descripción)
notificación	La instancia se está iniciando debido a que se supera el tiempo máximo permitido para estar detenida.
recuperación	Se ha iniciado la recuperación de la instancia . El tiempo de recuperación dependerá de la cantidad de datos que deban recuperarse.
recuperación	Ha finalizado la recuperación de la instancia.
Creación de parches de seguridad	La actualización del sistema operativo está disponible para su instancia. Para obtener más información acerca de cómo se aplican las actualizaciones, consulte Mantenimiento de Amazon DocumentDB.

Eventos de Amazon DocumentDB que se originan en un clúster

Categoría	Description (Descripción)
creación	Se ha creado un clúster
eliminación	Se ha eliminado un clúster.
conmutación por error	Volver a promocionar el principal anterior.
conmutación por error	Se ha completado la conmutación por error en la instancia.
conmutación por error	Se ha iniciado la misma conmutación por error de AZ en la instancia de base de datos: %s
conmutación por error	Se ha iniciado la misma conmutación por error de AZ en la instancia de base de datos: %s

Categoría	Description (Descripción)
conmutación por error	Se ha iniciado la conmutación por error entre AZ en la instancia de base de datos: %s
mantenimiento	Se ha parcheado el clúster.
mantenimiento	El clúster de base de datos tiene un estado que no se puede actualizar: %s
notificación	El clúster se ha detenido.
notificación	El clúster se ha iniciado.
notificación	La detención del clúster ha producido un error.
notificación	El clúster se está iniciando debido a que se supera el tiempo máximo permitido para estar detenida.
notificación	Se ha cambiado el nombre del clúster de %s a %s.

Eventos de Amazon DocumentDB que se originan en una instantánea de clúster

En la siguiente tabla se muestra la categoría de eventos y una lista de los eventos que pueden producirse cuando el tipo de origen es una instantánea de clúster de Amazon DocumentDB.

Categoría	Description (Descripción)
copia de seguridad	Crear una instantánea manual del clúster.
copia de seguridad	Se ha creado una instantánea manual del clúster.
copia de seguridad	Creación de instantáneas de clúster automatiz adas.

Categoría	Description (Descripción)
copia de seguridad	Se ha creado una instantánea de clúster automatizada.

Eventos de Amazon DocumentDB que se originan en un grupo de parámetros

En la siguiente tabla, se muestra la categoría de eventos y una lista de eventos correspondiente a un grupo de parámetros como tipo de origen.

Categoría	Description (Descripción)
cambio de configuración	Se actualizó el parámetro %s a %s con el método de aplicación %s

Supervisión de Amazon DocumentDB con CloudWatch

Amazon DocumentDB (compatible con MongoDB) se integra con Amazon CloudWatch para que pueda recopilar y analizar las métricas operativas de sus clústeres. Puede supervisar estas métricas mediante la CloudWatch consola, la consola de Amazon DocumentDB, la AWS Command Line Interface (AWS CLI) o la API. CloudWatch

CloudWatch también le permite configurar alarmas para que se le notifique si un valor métrico supera el umbral que especifique. Incluso puedes configurar Amazon CloudWatch Events para que tome medidas correctivas en caso de que se produzca una infracción. Para obtener más información sobre el uso CloudWatch y las alarmas, consulta la CloudWatch documentación de Amazon.

Temas

- Métricas de Amazon DocumentDB
- Visualización CloudWatch de datos
- Dimensiones de Amazon DocumentDB
- Supervisión de las métricas de opcounter
- Supervisión de conexiones a bases de datos

Métricas de Amazon DocumentDB

Para monitorizar el estado y el rendimiento del clúster e instancias de Amazon DocumentDB, puede consultar las siguientes métricas en la consola de Amazon DocumentDB.

> **Note**
>
> Las métricas de las siguientes tablas se aplican tanto a los clústeres elásticos como a los basados en instancias.

Temas

- Métricas de utilización de recursos
- Métricas de latencia
- NVMe-backed métricas de instancia
- Métricas operativas
- Command-level métricas
- Métricas de rendimiento
- Métricas del sistema
- Métricas de la instancia T3

Métricas de utilización de recursos

Métrica	Description (Descripción)	
`BackupRetentionPeriodStorageUsed`	La cantidad total de almacenamiento de copias de seguridad en bytes utilizada para admitir la característica de restauración a un momento dado dentro del período de retención de Amazon DocumentDB. Se incluye en el total registrado por la métrica `TotalBack`	

Métrica	Description (Descripción)	
	upStorageBilled . Se calcula de forma independi ente para cada clúster de Amazon DocumentDB.	
ChangeStreamLogSize	La cantidad de almacenam iento que utiliza el clúster para almacenar el registro de flujos de cambios en megabytes. Este valor es un subconjunto del almacenamiento total del clúster (VolumeBytesUsed) y afecta al costo del clúster. Para obtener información acerca de los precios de almacenamiento, consulte la página del producto de Amazon DocumentDB. El tamaño del registro de flujos de cambios es una función que indica la cantidad de cambios que se realizan en el clúster y el periodo en el que se conservará dicho registro de flujos de cambios. Para obtener más informaci ón acerca de las secuencias de cambio, consulte Uso de flujos de cambios con Amazon DocumentDB.	
CPUUtilization	Porcentaje de CPU usado por una instancia.	

Métrica	Description (Descripción)	
`DatabaseConnections`	El número de conexiones abiertas (activas e inactivas) en una instancia tomada en una frecuencia de 1 minuto.	
`DatabaseConnectionsMax`	El número máximo de conexiones abiertas (activas e inactivas) en una instancia en un período de 1 minuto.	
`DatabaseConnectionsLimit`	El número máximo de conexiones de bases de datos simultáneas (activas e inactivas) permitidas en una instancia en un momento dado.	
`DatabaseConnectionsActiveMax`	El número máximo de conexiones de base de datos activas en una instancia en un período de 1 minuto. Las conexiones activas son las que están ejecutando una operación en ese momento, a diferencia de las conexiones inactivas. Para conocer las cuotas de conexión activas por tipo de instancia, consulteCuotas de instancias.	
`DatabaseCursors`	El número de cursores abiertos en una instancia tomados en una frecuencia de 1 minuto.	

Métrica	Description (Descripción)
DatabaseCursorsMax	El número máximo de cursores abiertos en una instancia en un período de 1 minuto.
DatabaseCursorsLimit	El número máximo de cursores permitidos en una instancia en un momento dado.
DatabaseCursorsTimedOut	El número de cursores cuyo tiempo de espera se agotó en un período de 1 minuto.
FreeableMemory	Cantidad de memoria de acceso aleatorio disponible en bytes.
FreeLocalStorage	Esta métrica indica la cantidad de almacenamiento disponible en cada instancia para las tablas y los registros temporales. Este valor depende de la clase de instancia. Puede aumentar la cantidad de espacio de almacenamiento libre para una instancia eligiendo una clase de instancia más grande para ella. (Esto no se aplica a DocumentDB Serverless).

Métrica	Description (Descripción)
`LowMemThrottleQueueDepth`	La profundidad de la cola para las solicitudes que están limitadas debido a la poca memoria disponible tomada en una frecuencia de 1 minuto.
`LowMemThrottleMaxQueueDepth`	La profundidad máxima de la cola para las solicitudes que están limitadas debido a la poca memoria disponible en un período de 1 minuto.
`LowMemNumOperationsThrottled`	El número de solicitudes que están limitadas debido a la poca memoria disponible en un período de 1 minuto.
`SnapshotStorageUsed`	La cantidad total de almacenamiento de copias de seguridad en bytes consumida por todas las instantáneas de un clúster de Amazon DocumentDB determinado fuera de su período de retención de la copia de seguridad. Se incluye en el total registrado por la métrica `TotalBackupStorageBilled` . Se calcula de forma independiente para cada clúster de Amazon DocumentDB.

Métrica	Description (Descripción)
SwapUsage	La cantidad de espacio de intercambio utilizado en la instancia.
TotalBackupStorageBilled	La cantidad total de almacenamiento de copias de seguridad en bytes facturada para un clúster de Amazon DocumentDB determinado. Incluye el almacenamiento de copias de seguridad medido por las métricas BackupRetentionPeriodStorageUsed y SnapshotStorageUsed . Se calcula de forma independiente para cada clúster de Amazon DocumentDB.
TransactionsOpen	El número de transacciones abiertas en una instancia en una frecuencia de 1 minuto.
TransactionsOpenMax	El número máximo de transacciones abiertas en una instancia en un período de 1 minuto.
TransactionsOpenLimit	El número máximo de transacciones simultáneas permitidas en una instancia en un momento dado.

Métrica	Description (Descripción)
`VolumeBytesUsed`	Cantidad de almacenamiento utilizada por el clúster en bytes. Este valor afecta al costo del clúster. Para obtener información acerca de los precios, consulte la página del producto de Amazon DocumentDB.

Métricas de latencia

Métrica	Description (Descripción)
`DBClusterReplicaLagMaximum`	Retardo máximo en milisegundos entre la instancia principal y cada instancia de base de datos de Amazon DocumentDB del clúster.
`DBClusterReplicaLagMinimum`	Retardo mínimo en milisegundos entre la instancia principal y cada instancia de réplica del clúster.
`DBInstanceReplicaLag`	La cantidad de retardo, en milisegundos, cuando la replicación actualiza desde la instancia principal a una instancia de réplica.
`ReadLatency`	La cantidad media de tiempo empleado por operación de disco I/O .

Métrica	Description (Descripción)
`WriteLatency`	La cantidad media de tiempo, en milisegundos, que tarda cada I/O operación de disco.

NVMe-backed métricas de instancia

Métrica	Description (Descripción)
`NVMeStorageCacheHitRatio`	El porcentaje de solicitudes que se responden desde la caché por niveles.
`FreeNVMeStorage`	La cantidad de almacenamiento de NVMe efímero disponible.
`ReadIOPSNVMeStorage`	El número promedio de I/O operaciones de lectura de discos en un almacenamiento NVMe efímero.
`ReadLatencyNVMeStorage`	La cantidad media de tiempo empleado por I/O operación de lectura de disco para el almacenamiento efímero de NVMe.
`ReadThroughputNVMeStorage`	Número promedio de bytes leídos en el disco por segundo para el almacenamiento NVMe efímero.
`WriteIOPSNVMeStorage`	El número promedio de I/O operaciones de escritura en

Métrica	Description (Descripción)
	disco en un almacenamiento NVMe efímero.
`WriteLatencyNVMeStorage`	La cantidad media de tiempo empleado por I/O operación de escritura en disco para el almacenamiento efímero de NVMe.
`WriteThroughputNVMeStorage`	Número promedio de bytes escritos en el disco por segundo para el almacenamiento NVMe efímero.

Métricas operativas

Métrica	Description (Descripción)
`DocumentsDeleted`	El número de documentos eliminados en un período de 1 minuto.
`DocumentsInserted`	El número de documentos insertados en un período de 1 minuto.
`DocumentsReturned`	El número de documentos devueltos en un período de 1 minuto.
`DocumentsUpdated`	El número de documentos actualizados en un período de 1 minuto.

Métrica	Description (Descripción)
OpcountersCommand	El número de comandos emitidos en un período de 1 minuto.
OpcountersDelete	El número de operaciones de eliminación emitidas en un período de 1 minuto.
OpcountersGetmore	El número de getmores emitidos en un período de 1 minuto.
OpcountersInsert	El número de operaciones de inserción emitidas en un período de 1 minuto.
OpcountersQuery	El número de consultas emitidas en un período de 1 minuto.
OpcountersUpdate	El número de operaciones de actualización emitidas en un período de 1 minuto.
TransactionsStarted	El número de transacciones iniciadas en una instancia en un período de 1 minuto.
TransactionsCommitted	El número de transacciones realizadas en una instancia en un período de 1 minuto.
TransactionsAborted	El número de transacciones canceladas en una instancia en un período de 1 minuto.

Métrica	Description (Descripción)	
`TTLDeletedDocuments`	El número de documentos eliminados por un TTLMonitor en un período de 1 minuto.	

Command-level métricas

Amazon DocumentDB publica las métricas de latencia y concurrencia por comando para. CloudWatch Estas métricas le ayudan a identificar qué operaciones contribuyen más a su carga de trabajo y a detectar problemas de latencia o presión de concurrencia.

Cada comando de la tabla siguiente emite tres métricas en un período de 1 minuto:

- `AvgDuration`— El tiempo medio de ejecución del comando, en microsegundos.
- `P100Duration`— El tiempo máximo de ejecución del comando (percentil 100), en microsegundos.
- `MaxConcurrent`— El número máximo de ejecuciones simultáneas del comando.

Los nombres de las métricas siguen el patrón. `Command.`*`commandName`*`.`*`metricName`* Por ejemplo, `Command.find.AvgDuration` o `Command.aggregate.MaxConcurrent`.

Métrica	Description (Descripción)
`Command.find`	Lea las consultas mediante la función de búsqueda.
`Command.insert`	Operaciones de inserción de documentos.
`Command.findAndModify`	Operaciones atómicas de búsqueda y modificación.
`Command.update`	Operaciones de actualización de documentos.
`Command.delete`	Operaciones de eliminación de documentos.
`Command.aggregate`	Ejecuciones de canalizaciones de agregación.
`Command.count`	Operaciones de recuento.

Métrica	Description (Descripción)
`Command.distinct`	Consultas de valores distintos.
`Command.getMore`	Operaciones de recuperación por lotes de cursores.
`Command.abortTransaction`	Operaciones de anulación de transacciones.
`Command.commitTransaction`	Operaciones de confirmación de transacciones.

Métricas de rendimiento

Métrica	Description (Descripción)
`NetworkReceiveThroughput`	Cantidad de rendimiento de red en bytes por segundo recibida de los clientes por cada instancia del clúster. Este rendimiento no incluye el tráfico de red entre las instancias del clúster y el volumen del clúster.
`NetworkThroughput`	Cantidad de rendimiento de red en bytes por segundo recibida de los clientes y transmitida a ellos por cada instancia del clúster de Amazon DocumentDB. Este rendimiento no incluye el tráfico de red entre las instancias del clúster y el volumen del clúster.

Métrica	Description (Descripción)	
NetworkTransmitThroughput	Cantidad de rendimiento de red en bytes por segundo enviada a los clientes por cada instancia del clúster de base de datos. Este rendimiento no incluye el tráfico de red entre las instancias del clúster y el volumen del clúster.	
ReadIOPS	La cantidad promedio de I/O operaciones de lectura de discos por segundo. Amazon DocumentDB indica las IOPS de lectura y escritura por separado, en intervalos de 1 minuto.	
ReadThroughput	El número medio de bytes leídos del disco por segundo.	
StorageNetworkReceiveThroughput	Cantidad de rendimiento de red en bytes por segundo recibida del volumen de almacenamiento del clúster de Amazon DocumentDB por cada instancia del clúster.	
StorageNetworkTransmitThroughput	Cantidad de rendimiento de red en bytes por segundo enviada al volumen de almacenamiento del clúster de Amazon DocumentDB por cada instancia del clúster.	

Métrica	Description (Descripción)	
`StorageNetworkThroughput`	Cantidad de rendimiento de red en bytes por segundo recibida y enviada al volumen de almacenamiento del clúster de Amazon DocumentDB por cada instancia del clúster de Amazon DocumentDB.	

Métrica	Description (Descripción)	
`VolumeReadIOPs`	El número promedio de I/O operaciones de lectura facturadas de un volumen de clúster, registrado en intervalos de 5 minutos. Las operaciones de lectura facturadas se calculan en el nivel del volumen de clúster, se agrupan para todas las instancias del clúster y se notifican a intervalos de 5 minutos. El valor se calcula tomando el valor de la métrica de operaciones de lectura a lo largo de un periodo de 5 minutos. Puede determina r la cantidad de operacion es de lectura facturadas por segundo tomando el valor de la métrica de operaciones de lectura facturadas y dividiénd ola por 300 segundos. Por ejemplo, si las `VolumeReadIOPs` devuelven 13 686, las operaciones de lectura facturadas por segundo serán 45 (13 686/300 = 45,62). Las operaciones de lectura facturadas se acumulan para las consultas que solicitan páginas de la base de datos que no están presentes en	

Métrica	Description (Descripción)	
	la caché del búfer y que por tanto se deben cargar desde el almacenamiento. Es posible que aparezcan picos en las operaciones de lectura facturadas, ya que los resultados de la consulta se leen desde el almacenamiento y se cargan en la caché del búfer.	

Métrica	Description (Descripción)	
`VolumeWriteIOPs`	El número promedio de I/O operaciones de escritura facturadas de un volumen de clúster, registrado en intervalos de 5 minutos. Las operaciones de escritura facturadas se calculan en el nivel del volumen de clúster, se agrupan para todas las instancias del clúster y se notifican a intervalos de 5 minutos. El valor se calcula tomando el valor de la métrica de operaciones de escritura a lo largo de un periodo de 5 minutos. Puede determina r la cantidad de operaciones de escritura facturadas por segundo tomando el valor de la métrica de operacion es de escritura facturadas y dividiendo por 300 segundos. Por ejemplo, si las `VolumeWriteIOPs` devuelven 13 686, las operaciones de escritura facturadas por segundo serán 45 (13 686/300 = 45,62). Tenga en cuenta que las métricas `VolumeReadIOPs` y `VolumeWriteIOPs` se calculan mediante la capa de almacenamiento	

Métrica	Description (Descripción)	
	de DocumentDB e incluyen las iOS realizadas por las instancias principal y de réplica. Los datos se agregan cada 20 a 30 minutos y después, se generan informes en intervalos de 5 minutos, por lo que se emite el mismo punto de datos para la métrica en el período de tiempo. Si busca una métrica que se correlaci one con tus operaciones de inserción en un intervalo de 1 minuto, puede usar la métrica WriteIOps a nivel de instancia . La métrica está disponible en la pestaña de supervisión de la instancia principal de Amazon DocumentDB.	
`WriteIOPS`	La cantidad promedio de I/O operaciones de escritura en disco por segundo. Cuando se utilizan a nivel de clúster, `WriteIOPs` se evalúan en todas las instancias del clúster. Las IOPS de lectura y escritura se registran por separado, en intervalos de 1 minuto.	
`WriteThroughput`	Número medio de bytes que se escriben en el disco por segundo.	

Métricas del sistema

Métrica	Description (Descripción)	
`AvailableMVCCIds`	Un contador que muestra el número de operaciones de escritura restantes disponibl es antes de llegar a cero. Cuando este contador llegue a cero, el clúster pasará al modo de solo lectura hasta que se recuperen y reciclen los ID. El contador disminuye con cada operación de escritura y aumenta a medida que la recopilación de elementos no utilizados recicla los identific adores de MVCC antiguos.	
`BufferCacheHitRatio`	Porcentaje de solicitudes que se responden desde la caché de búfer.	
`DiskQueueDepth`	El número de I/O operacion es que están esperando a escribirse en el disco o leerse desde él.	
`EngineUptime`	Cantidad de tiempo en segundos que la instancia lleva en ejecución.	
`IndexBufferCacheHi tRatio`	Porcentaje de solicitudes de índice que se responden desde la caché de búfer. Es posible que vea un pico superior al 100 % en la métrica justo después	

Métrica	Description (Descripción)	
	de eliminar un índice, una colección o una base de datos. Esto se resolverá automáticamente después de 60 segundos. Esta limitación se corregirá en una actualización futura del parche.	
`LongestActiveGCRuntime`	Duración en segundos del proceso activo de recopilación de elementos no utilizados más prolongado. Se actualiza cada minuto y realiza un seguimiento únicamente de las operaciones activas, excluyendo los procesos que se completan en el intervalo de un minuto.	

Métricas de la instancia T3

Métrica	Description (Descripción)	
`CPUCreditUsage`	El número total de créditos de CPU que se han gastado durante el periodo de medición.	
`CPUCreditBalance`	El número de créditos de la CPU que ha acumulado una instancia. Este saldo se agota cuando la CPU realiza ráfagas y los créditos de CPU	

Métrica	Description (Descripción)	
	se gastan más rápido de lo que se obtienen.	
`CPUSurplusCreditBalance`	La cantidad de créditos de CPU sobrantes gastados para mantener el rendimiento de la CPU cuando el CPUCredit Balance valor es cero.	
`CPUSurplusCreditsCharged`	El número de créditos de CPU sobrantes que superen la cantidad máxima de créditos de CPU que se pueden obtener en un periodo de 24 horas y que, por lo tanto, generan gastos adicionales. Para obtener más informaci ón, consulte Monitoreo de sus créditos CPU.	

Visualización CloudWatch de datos

Puede ver CloudWatch los datos de Amazon mediante la CloudWatch consola, la consola de Amazon DocumentDB AWS Command Line Interface (AWS CLI) o la CloudWatch API.

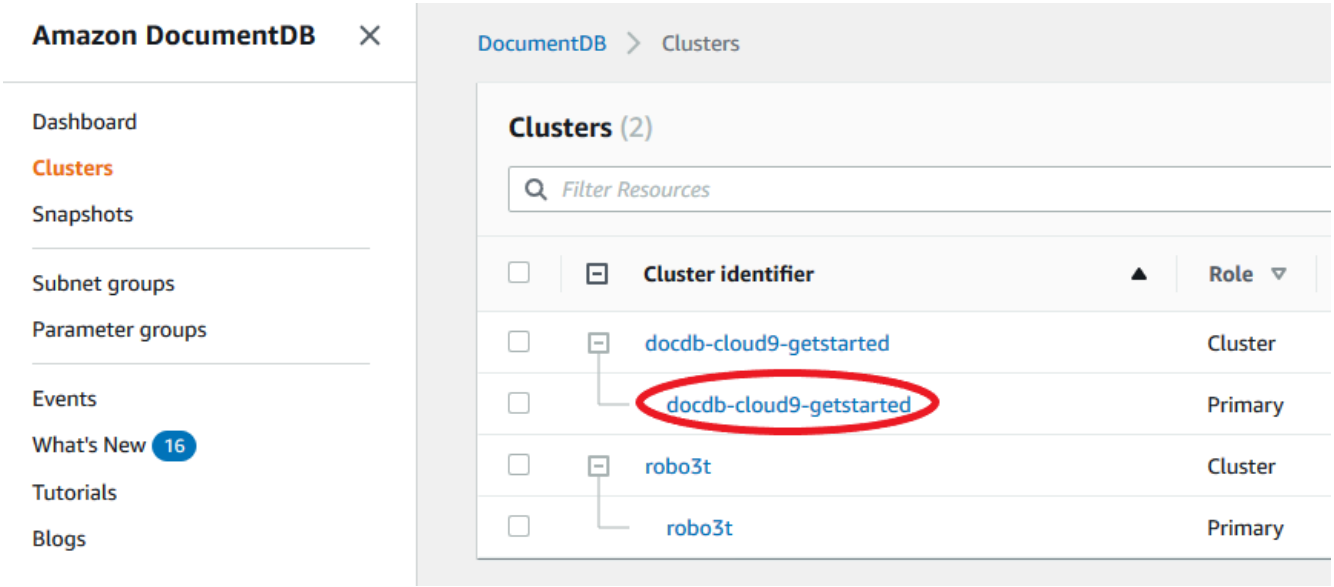
Using the Consola de administración de AWS

Para ver CloudWatch las métricas con la consola de administración de Amazon DocumentDB, complete los pasos siguientes.

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Clusters (Clústeres).

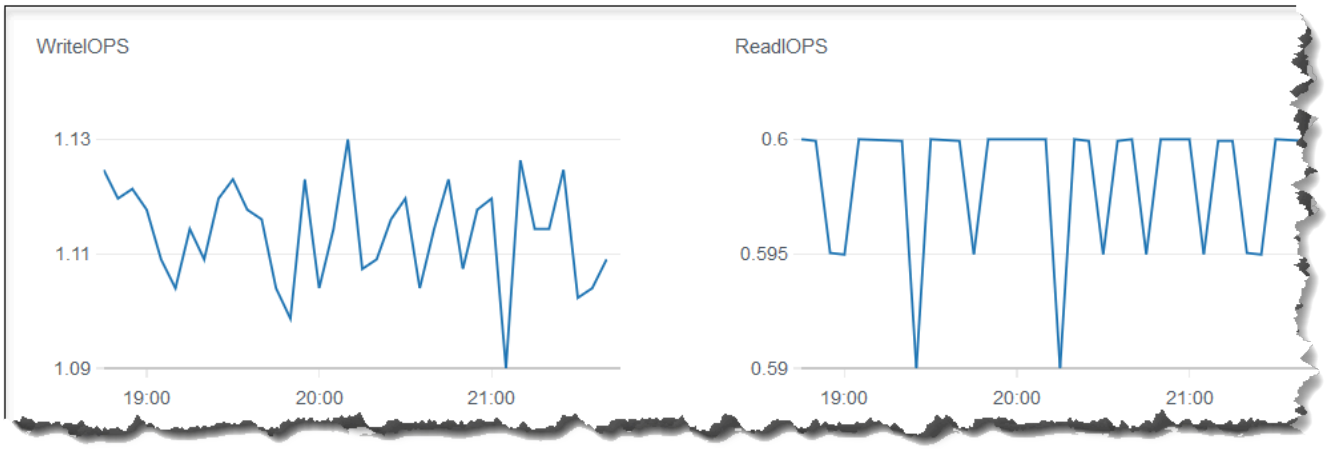
> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

3. En el cuadro de navegación de clústeres, verá la columna Identificador del clúster. Sus instancias se enumeran en clústeres, de forma similar a la siguiente captura de pantalla.



4. En la lista de instancias, seleccione el nombre de la instancia de la que desea ver las métricas.
5. En la página de resumen de instancias resultante, seleccione la pestaña Monitorización para ver las representaciones gráficas de las métricas de su instancia de Amazon DocumentDB. Como se debe generar un gráfico para cada métrica, es posible que los CloudWatch gráficos tarden unos minutos en completarse.

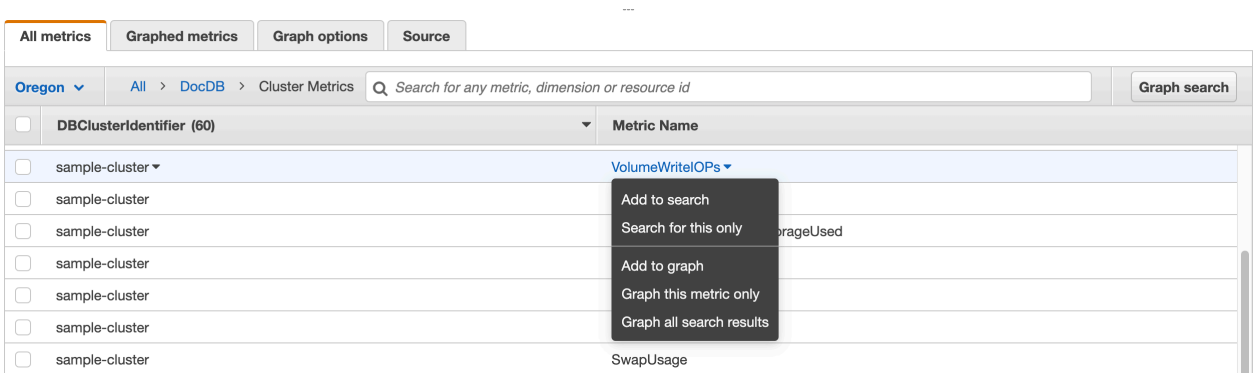
   La siguiente imagen muestra las representaciones gráficas de dos CloudWatch métricas en la consola de Amazon DocumentDB, y`WriteIOPS`. `ReadIOPS`



Using the CloudWatch Management Console

Para ver CloudWatch las métricas con la consola de CloudWatch administración, complete los pasos siguientes.

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB enhttps://console.aws.amazon.com/cloudwatch.
2. En el panel de navegación, seleccione Métricas. A continuación, en la lista de nombres de servicios, elija DocDB.
3. Elija una dimensión de métrica (por ejemplo, Métricas del clúster).
4. La pestaña Todas las métricas muestra todas las métricas para dicha dimensión en DocDB.
    a. Para ordenar la tabla, utilice el encabezado de columna.
    b. Para representar gráficamente una métrica, active la casilla de verificación situada junto a ella. Para seleccionar todas las métricas, seleccione la casilla de verificación en la fila de encabezado de la tabla.
    c. Para filtrar por métrica, coloque el cursor sobre el nombre de la métrica y seleccione la flecha desplegable situada junto al nombre de la métrica. A continuación, seleccione Añadir a la búsqueda, tal y como se muestra en la siguiente imagen.



Using the AWS CLI

Para ver CloudWatch los datos de Amazon DocumentDB, utilice la CloudWatch `get-metric-statistics` operación con los siguientes parámetros.

Parameters

- **`--namespace`**: obligatorio. El espacio de nombres del servicio para el que desea obtener métricas de CloudWatch . Para Amazon DocumentDB, debe ser `AWS/DocDB`.
- **`--metric-name`**: obligatorio. El nombre de la métrica para la que desea obtener datos.
- **`--start-time`**: obligatorio. La marca temporal que determina el primer punto de datos que se va a devolver.

  El valor especificado es inclusivo; los resultados incluyen puntos de datos con la marca temporal especificada. La marca de hora debe estar en formato ISO 8601 UTC (por ejemplo, 3 de octubre de 2016). T23:00:00Z
- **`--end-time`**: obligatorio. La marca temporal que determina el último punto de datos que se va a devolver.

  El valor especificado es inclusivo; los resultados incluyen puntos de datos con la marca temporal especificada. La marca de hora debe estar en formato ISO 8601 UTC (por ejemplo, 3 de octubre de 2016). T23:00:00Z
- **`--period`**: obligatorio. El grado de detalle, en segundos, de los puntos de datos devueltos. Para las métricas con una resolución normal, un periodo puede ser tan breve como un minuto (60 segundos) y debe ser un múltiplo de 60. Para las métricas de alta resolución que se recopilan a intervalos de menos de un minuto, el periodo puede ser 1, 5, 10, 30, 60 o cualquier múltiplo de 60.

- **`--dimensions`**— Opcional. Si la métrica contiene varias dimensiones, debes incluir un valor para cada dimensión. CloudWatch trata cada combinación única de dimensiones como una métrica independiente. No se pueden recuperar estadísticas utilizando combinaciones de dimensiones que no se han publicado. Debe especificar las mismas dimensiones que se utilizaron al crear las métricas.
- **`--statistics`**: opcional. La estadística de la métrica, distinta del percentil. Para la estadística de percentil, utilice `ExtendedStatistics`. Cuando llama a `GetMetricStatistics`, debe especificar `Statistics` o `ExtendedStatistics`, pero no ambos.

  Valores permitidos:

  - `SampleCount`
  - `Average`
  - `Sum`
  - `Minimum`
  - `Maximum`
- **`--extended-statistics`**: opcional. La estadística `percentile`. Especifique valores comprendidos entre p0.0 y p100. Cuando llama a `GetMetricStatistics`, debe especificar `Statistics` o `ExtendedStatistics`, pero no ambos.
- **`--unit`**: opcional. La unidad de una métrica determinada. Las métricas se pueden registrar en varias unidades. Si no se especifica una unida, se devuelven todas las unidades. Si especifica solo una unidad que la métrica no registra, los resultados de la llamada son nulos.

  Valores posibles:

  - `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

En el siguiente ejemplo se busca el valor máximo de `CPUUtilization` para un periodo de 2 horas tomando una muestra cada 60 segundos.

Para 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
```

Para 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
```

La salida de esta operación será similar a lo que se indica a continuación.

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

Dimensiones de Amazon DocumentDB

Las métricas de Amazon DocumentDB se identifican por los valores de la cuenta o de la operación. Puede usar la CloudWatch consola para recuperar los datos de Amazon DocumentDB filtrados por cualquiera de las dimensiones de la tabla siguiente.

Dimensión	Description (Descripción)
`DBClusterIdentifier`	Filtra los datos solicitados que son específicos del clúster de Amazon DocumentDB.
`DBClusterIdentifier, Role`	Filtra los datos que solicita para un clúster de Amazon DocumentDB específico y agrega la métrica por rol de instancia (). WRITER/RE ADER Por ejemplo, puede agregar métricas para todas las instancias READER que pertenezcan a un clúster.
`DBInstanceIdentifier`	Filtra los datos solicitados para una instancia de base de datos específica.

Supervisión de las métricas de opcounter

Las métricas de Opcounter tienen un valor distinto de cero (normalmente ~ 50) para los clústeres inactivos. Esto se debe a que Amazon DocumentDB realiza comprobaciones de estado periódicas, operaciones internas y tareas de recopilación de métricas.

Supervisión de conexiones a bases de datos

Al ver el número de conexiones mediante comandos del motor de base de datos`db.runCommand( { serverStatus: 1 })`, es posible que veas hasta 10 conexiones más de las que ves. `DatabaseConnections` CloudWatch Esto ocurre porque Amazon DocumentDB realiza comprobaciones de estado periódicas y tareas de recopilación de métricas que no se tienen en cuenta en `DatabaseConnections`. `DatabaseConnections` representa únicamente las conexiones iniciadas por el cliente.

Registro de llamadas a la API de Amazon DocumentDB con AWS CloudTrail

Amazon DocumentDB (compatible con MongoDB) está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por los usuarios o los roles o con un servicio de AWS Amazon DocumentDB (compatible con MongoDB). CloudTrail captura todas las llamadas a la AWS CLI API de Amazon DocumentDB como eventos, incluidas las llamadas desde la consola de Amazon DocumentDB y las llamadas de código al SDK de Amazon DocumentDB. Si crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos de Amazon DocumentDB. Si no configura una ruta, podrá ver los eventos más recientes en la CloudTrail consola, en el historial de eventos. Con la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a Amazon DocumentDB (con compatibilidad con MongoDB), la dirección IP desde la que se realizó la solicitud, quién la realizó, cuándo se realizó y otros detalles.

> **Important**
>
> Para determinadas funciones de administración, Amazon DocumentDB utiliza tecnología operativa que se comparte con Amazon Relational Database Service (Amazon RDS). Las llamadas a la consola y a la API de Amazon DocumentDB se registran como llamadas realizadas a la API de Amazon RDS. AWS CLI

Para obtener más información AWS CloudTrail, consulte la Guía AWS CloudTrail del usuario.

Información de Amazon DocumentDB en CloudTrail

CloudTrail está activado en su cuenta Cuenta de AWS al crear la cuenta. Cuando se produce una actividad en Amazon DocumentDB (compatible con MongoDB), esa actividad se registra en un CloudTrail evento junto con otros AWS eventos de servicio en el historial de eventos. Puede ver, buscar y descargar los eventos recientes en su. Cuenta de AWS Para obtener más información, consulte Visualización de eventos con el historial de CloudTrail eventos.

Para obtener un registro continuo de los eventos en su sitio Cuenta de AWS, incluidos los eventos de Amazon DocumentDB (con compatibilidad con MongoDB), cree un registro. Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. De forma predeterminada, cuando se crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las Regiones de AWS. El registro de seguimiento registra los eventos de todas las regiones de la

partición de AWS y envía los archivos de registro al bucket de Amazon S3 especificado. Además, puede configurar otros AWS servicios para analizar más a fondo los datos de eventos recopilados en los CloudTrail registros y actuar en función de ellos. Para obtener más información, consulte los siguientes temas en la Guía del usuario de AWS CloudTrail :

- Introducción a la creación de registros de seguimiento
- CloudTrail Integraciones y servicios compatibles
- Configuración de las notificaciones de Amazon SNS para CloudTrail
- Recibir archivos de CloudTrail registro de varias regiones
- Recibir archivos de CloudTrail registro de varias cuentas

Cada entrada de registro o evento incluye información acerca de quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con las credenciales raíz o del usuario.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro AWS servicio.

Para obtener más información, consulte el Elemento userIdentity de CloudTrail.

Creación de perfiles de operaciones en Amazon DocumentDB

Puede utilizar el generador de perfiles en Amazon DocumentDB (compatible con MongoDB) para registrar el tiempo de ejecución y los detalles de las operaciones realizadas en el clúster. El generador de perfiles es útil para monitorizar las operaciones más lentas del clúster para ayudarle a mejorar el rendimiento de las consultas individuales y el rendimiento general del clúster.

De forma predeterminada, la característica del generador de perfiles está deshabilitada. Cuando está habilitado, el generador de perfiles registra en Amazon Logs las operaciones que están tardando más que un valor de umbral definido por el cliente (por ejemplo, 100 ms). CloudWatch Los detalles registrados incluyen el comando con perfil, la hora, el resumen del plan y los metadatos del cliente. Una vez registradas las operaciones en CloudWatch Logs, puede usar CloudWatch Logs Insights para analizar, supervisar y archivar los datos de creación de perfiles de Amazon DocumentDB. En la sección Consultas comunes se proporcionan consultas comunes.

Cuando está habilitado, el generador de perfiles utiliza recursos adicionales en el clúster. Comience con un valor de umbral alto (por ejemplo, 500 ms) y baje gradualmente el valor para identificar las operaciones lentas. Comenzar con un valor umbral de 50 ms puede provocar problemas de rendimiento en el clúster para aplicaciones de alto rendimiento. El generador de perfiles está habilitado en el nivel de clúster y funciona en todas las instancias y bases de datos de un clúster. Amazon DocumentDB registra las operaciones en Amazon CloudWatch Logs haciendo el mejor esfuerzo posible.

Si bien Amazon DocumentDB no impone ningún cargo adicional por habilitar el generador de perfiles, se le cobran las tarifas estándar por el uso de los registros. CloudWatch Para obtener información sobre CloudWatch los precios de Logs, consulte los precios de Amazon. CloudWatch

Temas

- Operaciones admitidas
- Limitaciones
- Habilitación del generador de perfiles de Amazon DocumentDB
- Deshabilitación del generador de perfiles de Amazon DocumentDB
- Deshabilitación de la exportación de registros del generador de perfiles
- Acceso a los registros del generador de perfiles de Amazon DocumentDB
- Consultas comunes

Operaciones admitidas

El generador de perfiles de Amazon DocumentDB admite las siguientes operaciones:

- `aggregate`
- `count`
- `delete`
- `distinct`
- `find` (OP_QUERY y comando)
- `findAndModify`
- `insert`
- `update`

Limitaciones

El generador de perfiles de consultas lento solo puede emitir registros del generador de perfiles si todo el conjunto de resultados de la consulta puede caber en un lote y si el conjunto de resultados es inferior a 16 MB (tamaño máximo de BSON). Los conjuntos de resultados de más de 16 MB se dividen automáticamente en varios lotes.

La mayoría de los controladores o shells pueden establecer un tamaño de lote predeterminado pequeño. Puede especificar el tamaño del lote como parte de su consulta. Para capturar registros de consultas lentos, utilice un tamaño de lote que supere el tamaño del conjunto de resultados esperado. Si no está seguro del tamaño del conjunto de resultados o si varía, defina el tamaño del lote en un número grande (por ejemplo, 100 000).

Sin embargo, si se utiliza un tamaño de lote mayor, será necesario recuperar más resultados de la base de datos antes de enviar una respuesta al cliente. En el caso de algunas consultas, esto puede provocar demoras más prolongadas antes de obtener los resultados. Si no tiene previsto consumir todo el conjunto de resultados, es posible que gaste más en I/Os procesar la consulta y desechar el resultado.

Habilitación del generador de perfiles de Amazon DocumentDB

Habilitar el generador de perfiles en un clúster es un proceso de tres pasos. Asegúrese de completar todos los pasos o los registros de creación de perfiles no se enviarán a CloudWatch Logs. El generador de perfiles se establece en el nivel de clúster y se realiza en todas las bases de datos e instancias del clúster.

Habilitación del generador de perfiles en un clúster

1. Dado que no puede modificar un grupo de parámetros de clúster predeterminado, asegúrese de que dispone de un grupo de parámetros de clúster personalizado disponible. Para obtener más información, consulte Creación de grupos de parámetros de clúster de Amazon DocumentDB.
2. Con un grupo de parámetros de clúster personalizado disponible, modifique los siguientes parámetros: `profiler`, `profiler_threshold_ms` y `profiler_sampling_rate`. Para obtener más información, consulte Modificación de grupos de parámetros de clúster de Amazon DocumentDB.
3. Cree o modifique su clúster para usar el grupo de parámetros de clúster personalizado y para permitir la exportación de `profiler` CloudWatch registros a Logs.

En las siguientes secciones se muestra cómo implementar estos pasos mediante Consola de administración de AWS y AWS Command Line Interface (AWS CLI).

Using the Consola de administración de AWS

1. Antes de comenzar, cree un clúster de Amazon DocumentDB y un grupo de parámetros de clúster personalizado si aún no tiene uno. Para obtener más información, consulte Creación de grupos de parámetros de clúster de Amazon DocumentDB y Creación de un clúster de Amazon DocumentDB.
2. Utilice un grupo de parámetros de clúster personalizado disponible para modificar los siguientes parámetros. Para obtener más información, consulte Modificación de grupos de parámetros de clúster de Amazon DocumentDB.

   - `profiler`: habilita o deshabilita la creación de perfiles de consultas. Los valores permitidos son `enabled` y `disabled`. El valor predeterminado es `disabled`. Para habilitar la creación de perfiles, establezca el valor en `enabled`.
   - `profiler_threshold_ms`— Cuando `profiler` se establece en`enabled`, todos los comandos que están tardando más `profiler_threshold_ms` tiempo del registrado CloudWatch. Los valores permitidos son `[50-INT_MAX]`. El valor predeterminado es `100`.
   - `profiler_sampling_rate`: la fracción de las operaciones lentas que deben registrarse o crear un perfil. Los valores permitidos son `[0.0-1.0]`. El valor predeterminado es `1.0`.
3. Modifica tu clúster para usar el grupo de parámetros de clúster personalizado y configura las exportaciones de registros del generador de perfiles para que se publiquen en Amazon CloudWatch.

   a. En el panel de navegación, elija Clusters (Clústeres) para agregar el grupo de parámetros personalizado a un clúster.

   b. Elija el botón situado a la izquierda del nombre del clúster al que desea asociar el grupo de parámetros. Seleccione Actions (Acciones) y, a continuación, Modify (Modificar) para modificar el clúster.

   c. En Cluster options (Opciones de clúster), elija el grupo de parámetros personalizados del paso anterior para agregárselo al clúster.

   d. En Exportaciones de registros, selecciona los registros del generador de perfiles para publicarlos en Amazon. CloudWatch

   e. Elija Continue (Continuar) para ver un resumen de las modificaciones.

f. Después de verificar los cambios, puede aplicarlos inmediatamente o durante el siguiente período de mantenimiento en Scheduling of modifications (Programación de modificaciones).

g. Elija Modify cluster (Modificar clúster) para actualizar el clúster con el nuevo grupo de parámetros.

Using the AWS CLI

El siguiente procedimiento habilita el generador de perfiles en todas las operaciones admitidas para el clúster `sample-cluster`.

1. Antes de comenzar, asegúrese de tener un grupo de parámetros de clúster personalizado disponible; para ello, ejecute el siguiente comando y revise el resultado de un grupo de parámetros de clúster cuyo nombre no contenga `default` y cuya familia de grupos de parámetros sea `docdb3.6`. Si no tiene un grupo de parámetros de clúster distinto del predeterminado, consulte Creación de grupos de parámetros de clúster de Amazon DocumentDB.

```
aws docdb describe-db-cluster-parameter-groups \
    --query 'DBClusterParameterGroups[*].
[DBClusterParameterGroupName,DBParameterGroupFamily]'
```

En el siguiente resultado, solo `sample-parameter-group` cumple ambos criterios.

```
[
       [
           "default.docdb3.6",
           "docdb3.6"
       ],
       [
           "sample-parameter-group",
           "docdb3.6"
       ]
]
```

2. Utilizando el grupo de parámetros de clúster personalizado, modifique los siguientes parámetros:

- `profiler`: habilita o deshabilita la creación de perfiles de consultas. Los valores permitidos son `enabled` y `disabled`. El valor predeterminado es `disabled`. Para habilitar la creación de perfiles, establezca el valor en `enabled`.
- `profiler_threshold_ms`— Cuando `profiler` está configurado en`enabled`, todos los comandos tardan más `profiler_threshold_ms` tiempo del registrado. CloudWatch Los valores permitidos son `[50-INT_MAX]`. El valor predeterminado es `100`.
- `profiler_sampling_rate`: la fracción de las operaciones lentas que deben registrarse o crear un perfil. Los valores permitidos son `[0.0-1.0]`. El valor predeterminado es `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. Modifique el clúster de Amazon DocumentDB para que utilice el grupo de parámetros de clúster personalizado `sample-parameter-group` del paso anterior y establezca el parámetro `--enable-cloudwatch-logs-exports` a `profiler`.

   El código siguiente modifica el clúster `sample-cluster` para que utilice el `sample-parameter-group` del paso anterior y se añade `profiler` a las exportaciones de CloudWatch registros habilitadas.

   ```
   aws docdb modify-db-cluster \
          --db-cluster-identifier sample-cluster \
          --db-cluster-parameter-group-name sample-parameter-group \
          --cloudwatch-logs-export-configuration '{"EnableLogTypes":["profiler"]}'
   ```

   La salida de esta operación será similar a lo que se indica a continuación.

   ```
   {
       "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
    }
}
```

Deshabilitación del generador de perfiles de Amazon DocumentDB

Para deshabilitar el generador de perfiles, inhabilita tanto el `profiler` parámetro como la exportación de `profiler` registros a CloudWatch Logs.

Deshabilitación del generador de perfiles

Puede inhabilitar el `profiler` parámetro mediante la tecla Consola de administración de AWS o AWS CLI, de la siguiente manera.

Using the Consola de administración de AWS

El siguiente procedimiento utiliza el Consola de administración de AWS para deshabilitar Amazon DocumentDB. `profiler`

1. Inicie sesión en y abra Consola de administración de AWS la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Grupos de parámetros. A continuación, elija el nombre del grupo de parámetros de clúster en el que desea deshabilitar el generador de perfiles.
3. En la página Cluster parameters (Parámetros de clúster) resultante, seleccione el botón situado a la izquierda del parámetro `profiler` y elija Edit (Editar).
4. En el cuadro de diálogo Modify profiler (Modificar generador de perfiles), elija `disabled` en la lista.
5. Elija Modify cluster parameter (Modificar el parámetro de clúster).

Using the AWS CLI

Para deshabilitar `profiler` en un clúster mediante la AWS CLI, modifique el clúster tal y como se muestra a continuación.

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 ParameterName=profiler,ParameterValue=disabled,ApplyMethod=immediate
```

Deshabilitación de la exportación de registros del generador de perfiles

Puede deshabilitar la exportación de `profiler` registros a CloudWatch registros mediante el comando Consola de administración de AWS o AWS CLI, de la siguiente manera.

Using the Consola de administración de AWS

El siguiente procedimiento utiliza el Consola de administración de AWS para inhabilitar la exportación de registros de Amazon DocumentDB a CloudWatch.

1. Abra la consola de Amazon DocumentDB en. https://console.aws.amazon.com/docdb
2. En el panel de navegación, seleccione Clusters (Clústeres). Elija el botón situado a la izquierda del nombre del clúster para el que desea deshabilitar la exportación de registros.
3. En el menú Actions (Acciones), elija Modify (Modificar).
4. Desplácese hacia abajo hasta la sección Log exports (Exportaciones de registros) y desactive Profiler logs (Registros del generador de perfiles).
5. Elija Continuar.
6. Revise los cambios y, a continuación, elija cuándo desea que se aplique este cambio en su clúster:
   - Apply during the next scheduled maintenance window (Aplicar durante el siguiente periodo de mantenimiento programado)
   - Apply immediately (Aplicar inmediatamente)
7. Elija Modificar clúster.

Using the AWS CLI

El código siguiente modifica el clúster `sample-cluster` y desactiva CloudWatch los registros del generador de perfiles.

Example

Para Linux, macOS o Unix:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["profiler"]}'
```

Para Windows:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["profiler"]}'
```

La salida de esta operación será similar a lo que se indica a continuación.

```
{
    "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
    }
}
```


Acceso a los registros del generador de perfiles de Amazon DocumentDB

Sigue estos pasos para acceder a los registros de tu perfil en Amazon. CloudWatch

1. Abre la CloudWatch consola en https://console.aws.amazon.com/cloudwatch/.
2. Asegúrese de que se encuentra en la misma región que el clúster de Amazon DocumentDB.
3. En el panel de navegación, elija Logs (Registros).

4. Para encontrar los registros del generador de perfiles de su clúster, en la lista, elija `/aws/docdb/yourClusterName/profiler`.

   Los registros de perfil de cada una de las instancias están disponibles debajo de los nombres de instancia respectivos.

Consultas comunes

A continuación se muestran algunas consultas comunes que puede utilizar para analizar los comandos con perfil. Para obtener más información sobre CloudWatch Logs Insights, consulte Análisis de datos de registro con CloudWatch Logs Insights y consultas de muestra.

Obtención de las 10 operaciones más lentas en una colección especificada

```
filter ns="test.foo" | sort millis desc | limit 10
```

Obtención de todas las operaciones de actualización de una colección que tardaron más de 60 ms

```
filter millis > 60 and op = "update"
```

Obtención de las 10 operaciones más lentas del último mes

```
sort millis desc | limit 10
```

Obtener todas las consultas con un resumen de plan COLLSCAN

```
filter planSummary="COLLSCAN"
```

Supervisión con información sobre rendimiento

Información sobre rendimiento amplía las características de monitorización existentes de Amazon DocumentDB para ilustrar el desempeño de su clúster y le ayuda a analizar cualquier problema que le afecte. Con el panel de Información sobre rendimiento, puede visualizar la carga de la base de datos y filtrarla por esperas, instrucciones de consulta, hosts o usuarios.

> **Note**
>
> Performance Insights solo está disponible para los clústeres basados en instancias de Amazon DocumentDB 3.6, 4.0, 5.0 y 8.0.

¿Para qué sirve?

- Visualice el rendimiento de la base de datos: visualice la carga para determinar cuándo y dónde se encuentra la carga en la base de datos
- Determine la causa de la carga en la base de datos: determine qué consultas, hosts y aplicaciones contribuyen a la carga de la instancia
- Determine cuándo hay carga en su base de datos: amplíe el panel de Información sobre rendimiento para centrarse en eventos específicos o reduzca el tamaño para observar las tendencias a lo largo de un período de tiempo más amplio
- Alerta sobre la carga de la base de datos: acceda automáticamente a las nuevas métricas de carga de la base de datos, desde CloudWatch donde podrá supervisar las métricas de carga de la base de datos junto con otras métricas de Amazon DocumentDB y establecer alertas sobre ellas

¿Cuáles son las limitaciones de Información sobre rendimiento de Amazon DocumentDB?

- La información sobre el rendimiento en las regiones AWS GovCloud (US-East) y AWS GovCloud (US-West) no está disponible
- La Información sobre rendimiento para Amazon DocumentDB retiene hasta 7 días de datos de rendimiento
- Las consultas de más de 1024 bytes no se agregan en Información sobre rendimiento

Temas

- Conceptos de Información sobre rendimiento
- Activación y desactivación de Información sobre rendimiento
- Configuración de directivas de acceso para información sobre rendimiento
- Análisis de métricas mediante el panel de Información sobre rendimiento
- Recuperación de métricas con la API de Información sobre rendimiento
- CloudWatch Métricas de Amazon para obtener información sobre el rendimiento

- Métricas de contador para Información sobre rendimiento

Conceptos de Información sobre rendimiento

Temas

- Sesiones activas promedio
- Dimensiones
- Max vCPU

Sesiones activas promedio

La carga de base de datos mide el nivel de actividad en la base de datos. La métrica clave en Información sobre rendimiento es `DB Load`, que se recopila cada segundo. La unidad para la métrica `DBLoad` es el promedio de sesiones activas (AAS) de la instancia de Amazon DocumentDB.

Una sesión activa es una conexión que ha enviado trabajo a la instancia de Amazon DocumentDB y está esperando una respuesta. Por ejemplo, si envía una consulta a una instancia de Amazon DocumentDB, la sesión de base de datos estará activa mientras el motor de base de datos procesa la consulta.

Para obtener un promedio de sesiones activas, la Información sobre rendimiento muestrea el número de sesiones que ejecutan una consulta al mismo tiempo. El AAS es el total del número de sesiones dividido entre el total del número de muestras. En la tabla siguiente se muestran 5 ejemplos consecutivos de una consulta en ejecución.

Ejemplo	Número de sesiones que ejecutan una consulta	AAS	Cálculo
1	2	2	2 sesiones / 1 muestra
2	0	1	2 sesiones / 2 muestras
3	4	2	6 sesiones / 3 muestras

Ejemplo	Número de sesiones que ejecutan una consulta	AAS	Cálculo
4	0	1.5	6 sesiones / 4 muestras
5	4	2	10 sesiones / 5 muestras

En el ejemplo anterior, la carga de base de datos para el intervalo de tiempo de 1 a 5 es 2 AAS. Un aumento en la carga de base de datos significa que, en promedio, se están ejecutando más sesiones en la base de datos.

Dimensiones

La métrica `DB Load` es distinta de las demás métricas de series temporales porque puede desglosarla en subcomponentes llamados dimensiones. Las dimensiones son una especie de categorías de las diferentes características de la métrica `DB Load`. Cuando se diagnostican problemas de rendimiento, las dimensiones más útiles son los eventos de espera y top SQL.

estados de espera

Un evento de espera hace que una instrucción de consulta espere a que ocurra un evento específico antes de que pueda continuar ejecutándose. Por ejemplo, una instrucción de consulta podría esperar hasta que se desbloqueara un recurso bloqueado. Al combinar `DB Load` con eventos de espera, puede obtener una imagen completa del estado de la sesión. Estos son varios estados de espera de Amazon DocumentDB:

Estado de espera de Amazon DocumentDB	Descripción del estado de espera
Pestillo	El estado de espera de pestillo se produce cuando la sesión está esperando para paginar el conjunto de búferes. La entrada y salida frecuentes del conjunto de búferes puede producirse con mayor frecuencia cuando el sistema procesa consultas frecuentes de gran tamaño, escanea colecciones o cuando el

Estado de espera de Amazon DocumentDB	Descripción del estado de espera
	grupo de búferes es demasiado pequeño para gestionar todo el conjunto de trabajo.
CPU	El estado de espera de la CPU se produce cuando la sesión está esperando en la CPU.
CollectionLock	El estado de CollectionLock espera se produce cuando la sesión está a la espera de bloquear la recopilación. Estos eventos se producen cuando hay operaciones de DDL en la colección.
DocumentLock	El estado de DocumentLock espera se produce cuando la sesión espera para bloquear un documento. Un número elevado de escrituras simultáneas en el mismo documento contribui rá a aumentar los estados de DocumentLock espera en ese documento.
SystemLock	El estado de SystemLock espera se produce cuando la sesión está en espera en el sistema. Esto puede ocurrir cuando hay consultas frecuentes de larga duración, transacciones de larga duración o mucha simultaneidad en el sistema.
E/S	El estado de espera de la E/S se produce cuando la sesión está esperando en la E/S.
BufferLock	El estado de BufferLock espera se produce cuando la sesión espera bloquear una página compartida en el búfer. BufferLockLos estados de espera pueden prolongarse si otros procesos tienen los cursores abiertos en las páginas solicitadas.

Estado de espera de Amazon DocumentDB	Descripción del estado de espera
LowMemThrottle	El estado de LowMemThrottle espera se produce cuando la sesión está en espera debido a una gran presión de memoria en la instancia de Amazon DocumentDB. Si este estado persiste durante mucho tiempo, considere la posibilidad de escalar verticalm ente la instancia para proporcionar memoria adicional. Para obtener más información, consulte Recurso de origen.
BackgroundActivity	El estado de BackgroundActivity espera se produce cuando la sesión espera un proceso interno del sistema.
Otro	El otro estado de espera es un estado de espera interno. Si este estado persiste durante mucho tiempo, considere la posibilidad de finalizar esta consulta. Para obtener más información, consulte ¿Cómo puedo encontrar y terminar las consultas que tardan mucho en ejecutarse o se bloquean?

Consultas principales

Mientras que los eventos de espera muestran los cuellos de botella, las consultas principales indican qué consultas están contribuyendo más a la carga de la base de datos. Por ejemplo, es posible que, aunque haya muchas consultas ejecutándose actualmente en la base de datos, una de ellas consuma el 99 % de la carga de la base de datos. En este caso, es posible que la carga alta indique un problema con la consulta.

Max vCPU

En el panel, el gráfico de Carga de base de datos recopila, agrega y muestra información de la sesión. Para ver si las sesiones activas superan el máximo de la CPU, observe su relación con la línea Máximo de la CPU virtual. El valor Máximo de la vCPU se determina por el número de núcleos de vCPU (CPU virtual) de la instancia de Amazon DocumentDB.

Si la carga de base de datos suele estar por encima de la línea Máximo de la CPU virtual y el estado de espera principal es CPU, la CPU del sistema está sobrecargada. En este caso, quizá sea conveniente limitar las conexiones con la instancia, ajustar las consultas con una carga de CPU alta o pensar en la posibilidad de usar una clase de instancia de mayor tamaño. Si hay instancias altas y uniformes en cualquier estado de espera, eso indica que es posible que haya problemas de contención de recursos o cuellos de botella que hay que resolver. Esto puede ser así aunque la carga de base de datos no cruce la línea de Máximo de la CPU virtual.

Activación y desactivación de Información sobre rendimiento

Para poder usar Información sobre rendimiento, debe activarlo en su instancia de base de datos. Si lo necesita, podrá desactivarlo más adelante. La activación y desactivación de Información sobre rendimiento no provoca tiempos de inactividad, un reinicio o una conmutación por error.

El agente Información sobre rendimiento consume CPU y memoria limitadas en el host de base de datos. Cuando la carga de la base de datos es alta, el agente limita el impacto en el rendimiento mediante la recopilación de datos con menos frecuencia.

Activación de Información sobre rendimiento al crear un clúster

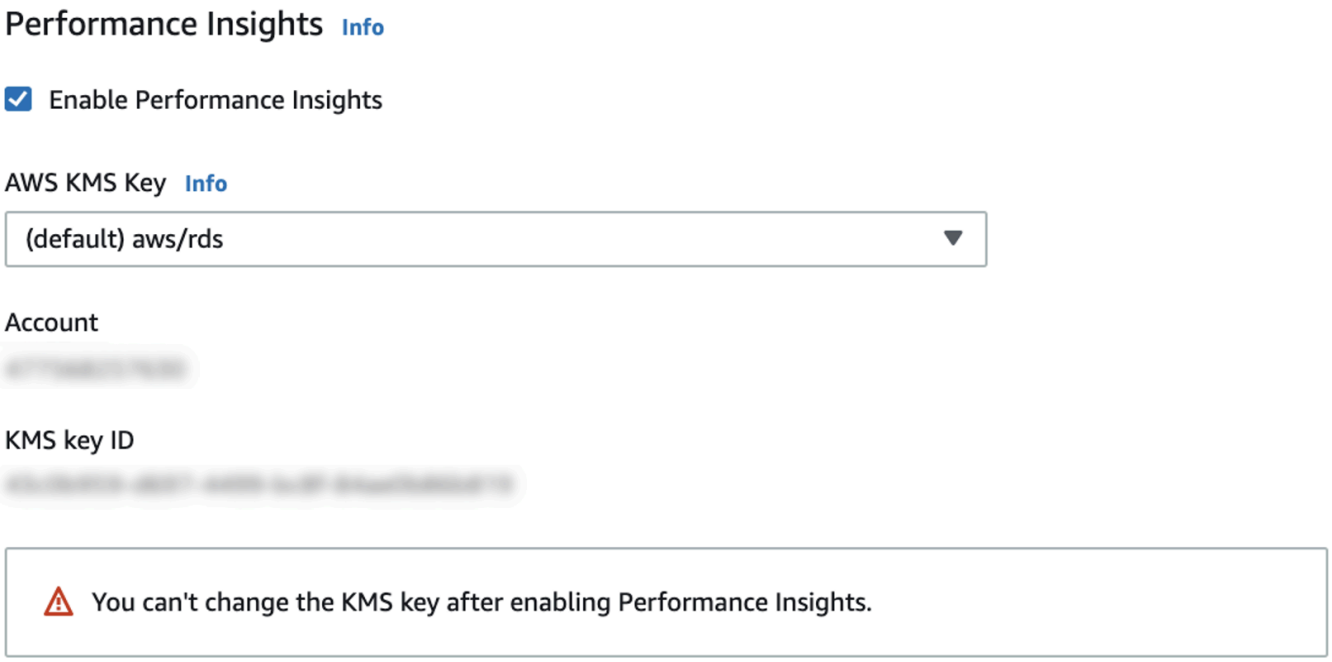
En la consola, puede activar o desactivar Información sobre rendimiento al crear o modificar una nueva instancia de base de datos.

Uso de Consola de administración de AWS

En la consola, puede activar o desactivar Información sobre rendimiento al crear un clúster de Amazon DocumentDB. Cuando cree un nuevo clúster de Amazon DocumentDB, active Información sobre rendimiento con la opción Habilitar Información sobre rendimiento de la sección Información sobre rendimiento.

Instrucciones de la consola

1. Para crear un clúster, siga las instrucciones para Creación de un clúster de base de datos de Amazon DocumentDB.
2. En la sección Información sobre rendimiento, elija Activar Información sobre rendimiento.



> **Note**
>
> El período de retención de datos de Información sobre rendimiento será de siete días.

AWS KMS clave: especifique su clave de AWS KMS. Performance Insights cifra todos los datos potencialmente confidenciales con su AWS KMS clave. Las datos se cifran en reposo y en tránsito. Para obtener más información, consulte Configurar una AWS AWS KMS política para Performance Insights.

Activación y desactivación al modificar una instancia

Puede modificar una instancia de base de datos para habilitar Información sobre rendimiento mediante la consola o la AWS CLI.

Using the Consola de administración de AWS

Instrucciones de la consola

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. Seleccione Clusters (Clústeres).

3. Seleccione una instancia de base de datos y elija Modificar.
4. En la sección Información sobre rendimiento, elija Activar Información sobre rendimiento o Desactivar Información sobre rendimiento.

> **Note**
>
> Si elige Enable Performance Insights, puede especificar su AWS AWS KMS clave. Performance Insights cifra todos los datos potencialmente confidenciales con su AWS KMS clave. Las datos se cifran en reposo y en tránsito. Para obtener más información, consulte Cifrado de datos de Amazon DocumentDB en reposo.

5. Elija Continuar.
6. En Programación de modificaciones, elija Aplicar inmediatamente. Si elige Aplicar durante la próxima ventana de mantenimiento programada, la instancia ignora esta configuración y habilita de inmediato Información sobre rendimiento.
7. Elija Modificar instancia.

Using the AWS CLI

Al usar los `modify-db-instance` AWS AWS CLI comandos `create-db-instance` o, puede habilitar Performance Insights `--enable-performance-insights` especificándolo o deshabilitarlo `--no-enable-performance-insights` especificándolo.

En el siguiente procedimiento, se describe cómo se activa o desactiva Información sobre rendimiento en una instancia de base de datos a través de AWS AWS CLI.

AWS AWS CLI instrucciones

Ejecute el `modify-db-instance` AWS AWS CLI comando y proporcione los valores siguientes:

- `--db-instance-identifer`: nombre de la instancia de la base de datos.
- `--enable-performance-insights` para activarlo o `--no-enable-performance-insights` para desactivarlo.

Example

El siguiente ejemplo habilita Información sobre rendimiento para `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
```

Configuración de directivas de acceso para información sobre rendimiento

Para acceder a Información sobre rendimiento, debe tener los permisos adecuados de AWS Identity and Access Management (IAM). Tiene las siguientes opciones para conceder acceso:

- Asocie la política administrada `AmazonRDSPerformanceInsightsReadOnly` a un conjunto de permisos o a un rol.
- Cree una política de IAM personalizada y asóciela a un conjunto de permisos o a un rol.

Además, si especificó una clave administrada por el cliente al activar Información sobre rendimiento, asegúrese de que los usuarios de su cuenta tengan los permisos `kms:Decrypt` y `kms:GenerateDataKey` sobre la clave de KMS.

> **Note**
>
> Para el cifrado inactivo con la administración de AWS KMS claves y grupos de seguridad, Amazon DocumentDB aprovecha la tecnología operativa que se comparte con Amazon RDS. https://aws.amazon.com/rds

Adjuntar la política a un principal de IAM AmazonRDSPerformanceInsightsReadOnly

`AmazonRDSPerformanceInsightsReadOnly`es una política AWS administrada que otorga acceso a todas las operaciones de solo lectura de la API Performance Insights de Amazon DocumentDB. Actualmente, todas las operaciones de esta API son de solo lectura. Si asocia

`AmazonRDSPerformanceInsightsReadOnly` a un conjunto de permisos o a un rol, el destinatario puede utilizar Información de rendimiento con las demás características de la consola.

Creación de una política de IAM personalizada para la información sobre rendimiento

Para los usuarios que no tienen la política `AmazonRDSPerformanceInsightsReadOnly`, puede conceder acceso a Información sobre rendimiento creando o modificando una política de IAM administrada por el usuario. Al asociar la política a un conjunto de permisos o a un rol, el destinatario puede utilizar Información de rendimiento.

Creación de una política personalizada

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación, seleccione Políticas.
3. Elija Create Policy (Crear política).
4. En la página Create Policy (Crear política), elija la pestaña JSON.
5. Copie y pegue el siguiente texto y *`us-east-1`* sustitúyalo por el nombre de su AWS región y por su número de cuenta de *`111122223333`* 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. Elija Review policy (Revisar política).
7. Proporcione un nombre para la política y, opcionalmente, una descripción, a continuación, elija Create policy (Crear política).

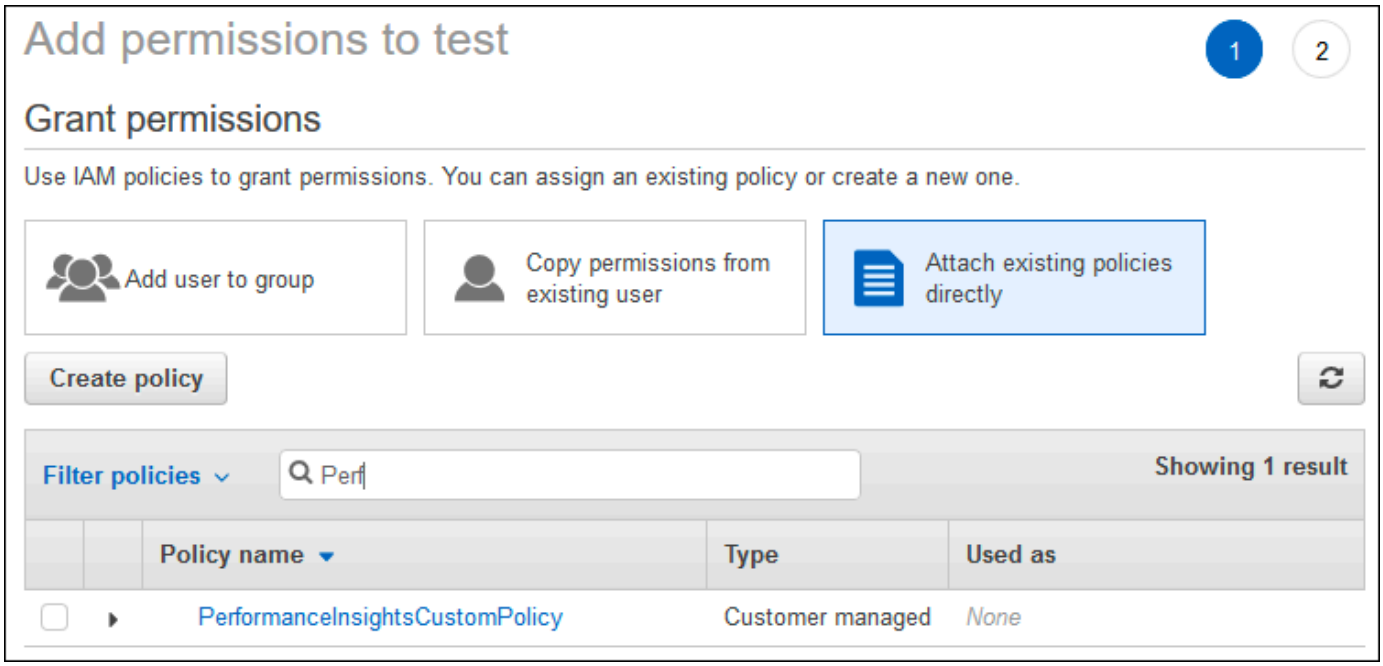
Ahora ya puede asociar la política a un conjunto de permisos o a un rol. En el procedimiento siguiente, se presupone que ya tiene un usuario para este fin.

Asociación de la política a un usuario

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación, seleccione Users.
3. Elija en la lista un usuario existente.

> **Important**
>
> Para utilizar Información sobre rendimiento, asegúrese de tener acceso a Amazon DocumentDB además de la política personalizada. Por ejemplo, la política AmazonDocDBReadOnlyAccess predefinida proporciona acceso de solo lectura a Amazon. Para obtener DocDB.For más información, consulte Gestión del acceso mediante políticas.

4. En la página Resumen, elija Añadir permisos.
5. Elija Attach existing policies directly (Adjuntar políticas existentes directamente). En Search (Buscar), escriba los primeros caracteres del nombre de la política, como se muestra más abajo.



6. Elija la política y, a continuación, elija Next: Review.
7. Elija Add permissions (Agregar permisos).

Configuración de un AWS KMS política de Performance Insights

Performance Insights utiliza un AWS KMS key para cifrar datos confidenciales. Cuando habilita la información sobre rendimiento a través de la API o la consola, tiene las siguientes opciones:

- Elija el valor predeterminado Clave administrada de AWS.

Amazon DocumentDB usa el Clave administrada de AWS para su nueva instancia de base de datos. Amazon DocumentDB crea una Clave administrada de AWS para su cuenta. AWS Su AWS cuenta de Amazon DocumentDB es diferente Clave administrada de AWS para cada AWS región.

- Elija una clave administrada por el cliente.

  Si especifica una clave administrada por el cliente, los usuarios de su cuenta que llamen a la API de Performance Insights necesitarán los permisos `kms:Decrypt` y `kms:GenerateDataKey` sobre la clave de KMS. Administre los permisos de la clave de KMS mediante la política de AWS KMS claves en lugar de utilizar las políticas de IAM. Para obtener más información, consulte Uso de políticas de claves en AWS KMS.

Example

La siguiente política de claves de ejemplo muestra cómo agregar instrucciones a la claves de KMS. Estas instrucciones permiten el acceso a la información sobre rendimiento. En función de cómo la utilices AWS KMS, es posible que desees cambiar algunas restricciones. Antes de agregar sentencias a la directiva, elimine todos los comentarios.

Análisis de métricas mediante el panel de Información sobre rendimiento

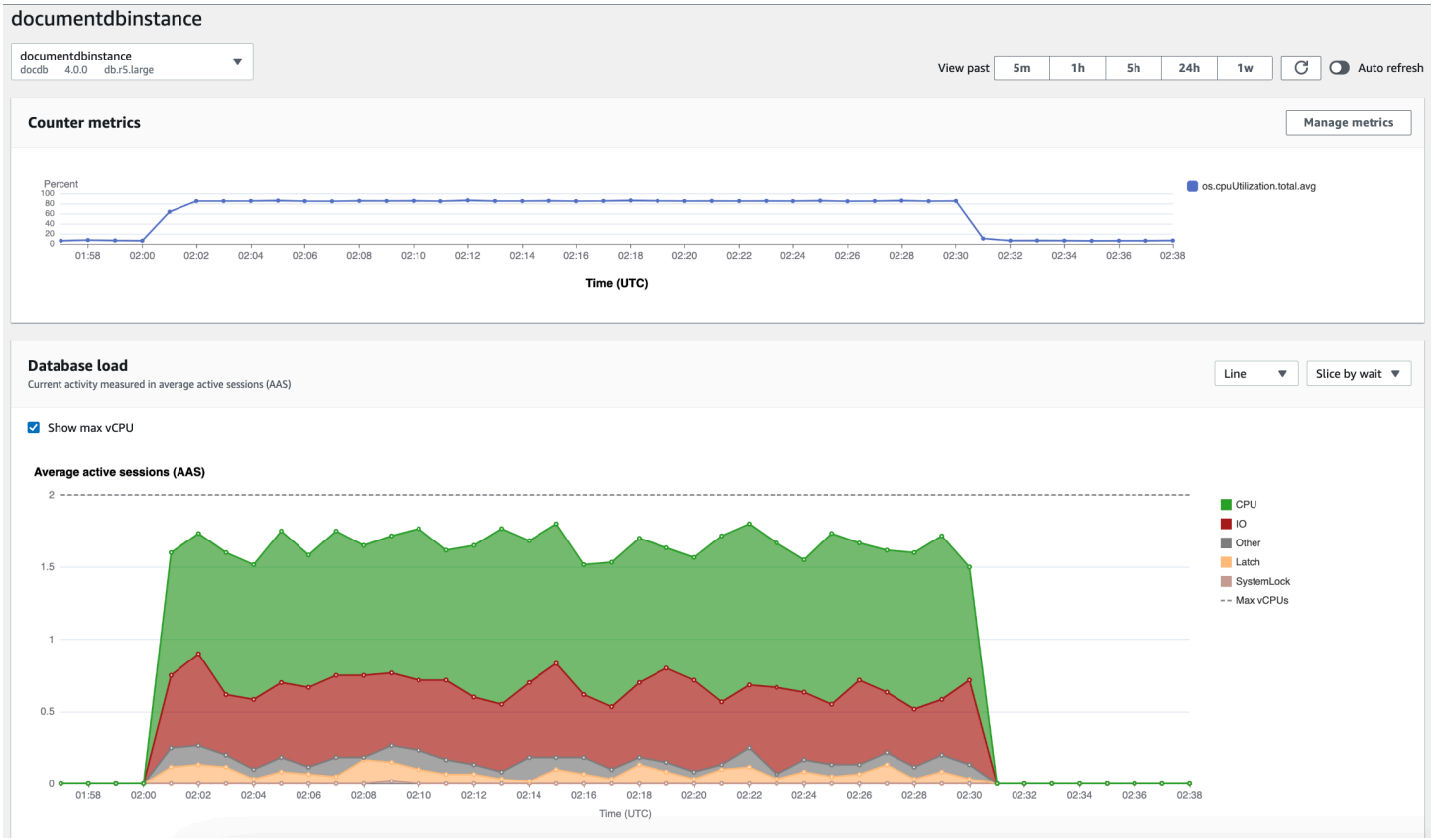
El panel de Información sobre rendimiento contiene información de desempeño de la base de datos para ayudarle a analizar y solucionar los problemas de desempeño. En la página del panel principal, encontrará información sobre la carga de la base de datos (DB load). Puede “dividir” la carga de la base de datos por dimensiones, como eventos de espera o de consulta.

Temas

- Información general del panel de Información sobre rendimiento
- Apertura del panel de Información sobre rendimiento
- Análisis de carga de la base de datos mediante estados de espera
- Información general sobre la pestaña Top queries
- Ampliación del gráfico de carga de base de datos

Información general del panel de Información sobre rendimiento

El panel es la forma más sencilla de interactuar con Información sobre rendimiento. El siguiente ejemplo muestra el panel de una instancia de Amazon DocumentDB. De forma predeterminada, el panel de Información sobre rendimiento muestra los datos de la última hora.



El panel está dividido en las partes siguientes:

1. Métricas de contador: muestra los datos para métricas de contador de rendimiento específicas.
2. Gráfico de carga de base de datos: muestra cómo se compara la carga de base de datos con la capacidad de instancia de base de datos representada por la línea de vCPU máximas.
3. Principales dimensiones: muestra las dimensiones principales que contribuyen a la carga de la base de datos. Estas dimensiones incluyen `waits`, `queries`, `hosts`, `databases` y `applications`.

Temas

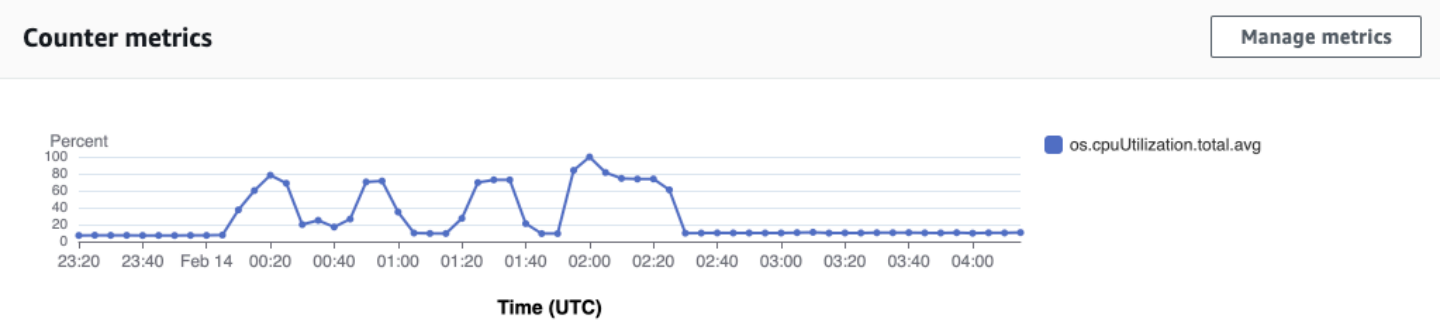
- Gráfico Counter metrics (Métricas de contador)
- Gráfico Database load (Carga de base de datos)

- Tabla de dimensiones principales

Gráfico Counter metrics (Métricas de contador)

Con las métricas de contador, puede personalizar el panel de Información sobre rendimiento para que incluya hasta 10 gráficos adicionales. Estos gráficos muestran una selección de docenas de métricas de rendimiento de sistemas operativos y bases de datos. Esta información se puede correlacionar con la carga de base de datos para ayudar a identificar y analizar problemas de rendimiento.

El gráfico Counter metrics (Métricas de contador) muestra los datos para los contadores de rendimiento.



Para cambiar los contadores de rendimiento, elija Administrar métricas. Puede seleccionar varias métricas del sistema operativo como se muestra en la siguiente captura de pantalla. Para ver los detalles de cualquier métrica, sitúe el cursor sobre el nombre de la métrica.

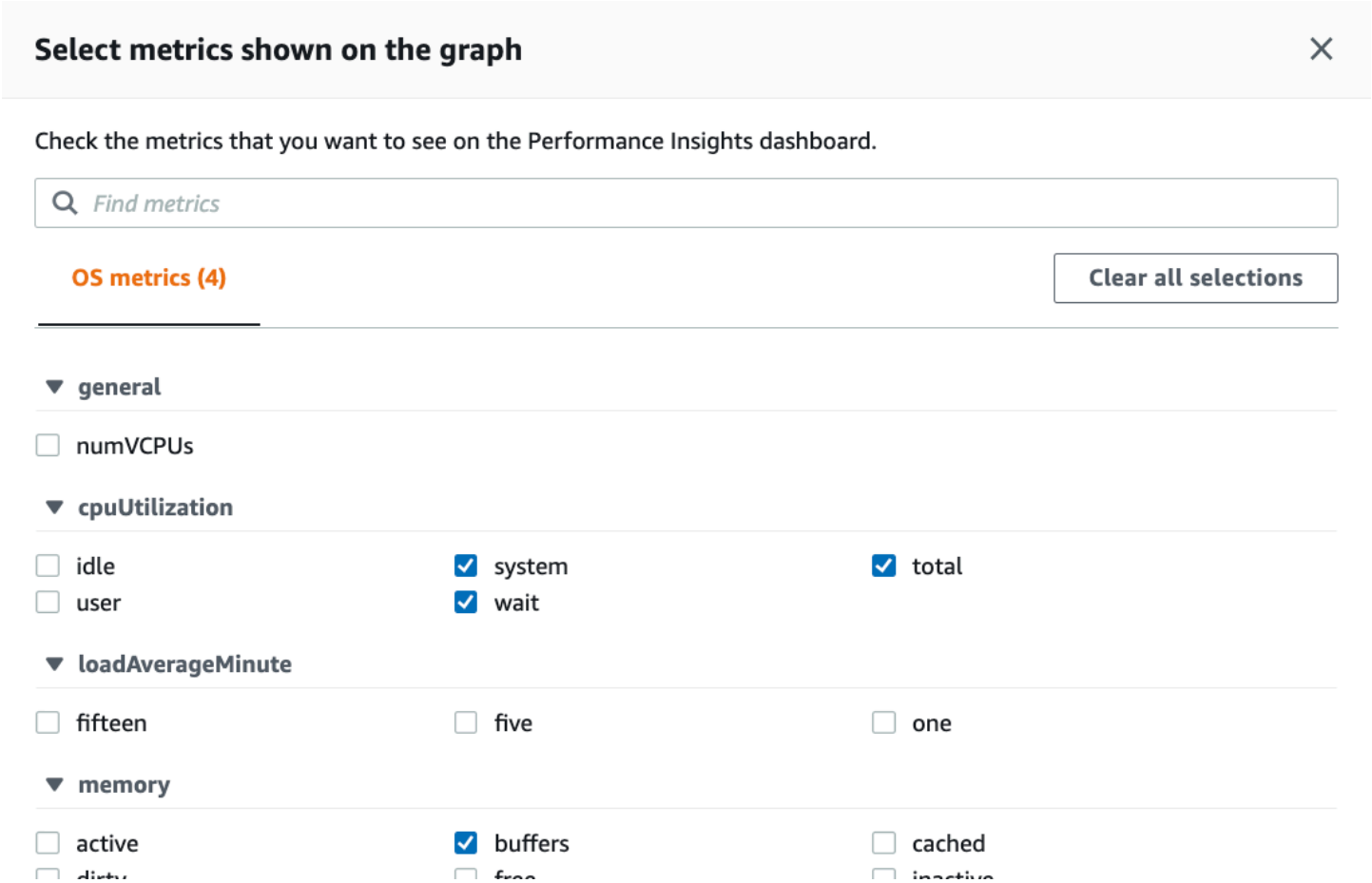


Gráfico Database load (Carga de base de datos)

El gráfico Carga de base de datos muestra cómo se compara la carga de base de datos con la capacidad de la instancia de base de datos representada por la línea Max vCPU (Máximo de vCPU). De forma predeterminada, el gráfico de líneas apilado representa la carga de base de datos como promedio de sesiones activas por unidad de tiempo. La carga de base de datos está dividida (agrupada) por estados de espera.



Carga de base de datos dividida por dimensiones

Puede elegir ver la carga como sesiones activas agrupadas por cualquier dimensión admitida. En la imagen siguiente, se muestran las dimensiones de una instancia de Amazon DocumentDB.



Detalles de carga de base de datos de un elemento de dimensión

Para consultar los detalles de un elemento de carga de base de datos dentro de una dimensión, pase el cursor sobre el nombre de elemento. En la imagen siguiente, se muestran los detalles de una instrucción de consulta.



Para consultar los detalles de cualquier elemento para el periodo de tiempo seleccionado en la leyenda, coloque el cursor sobre ese elemento.

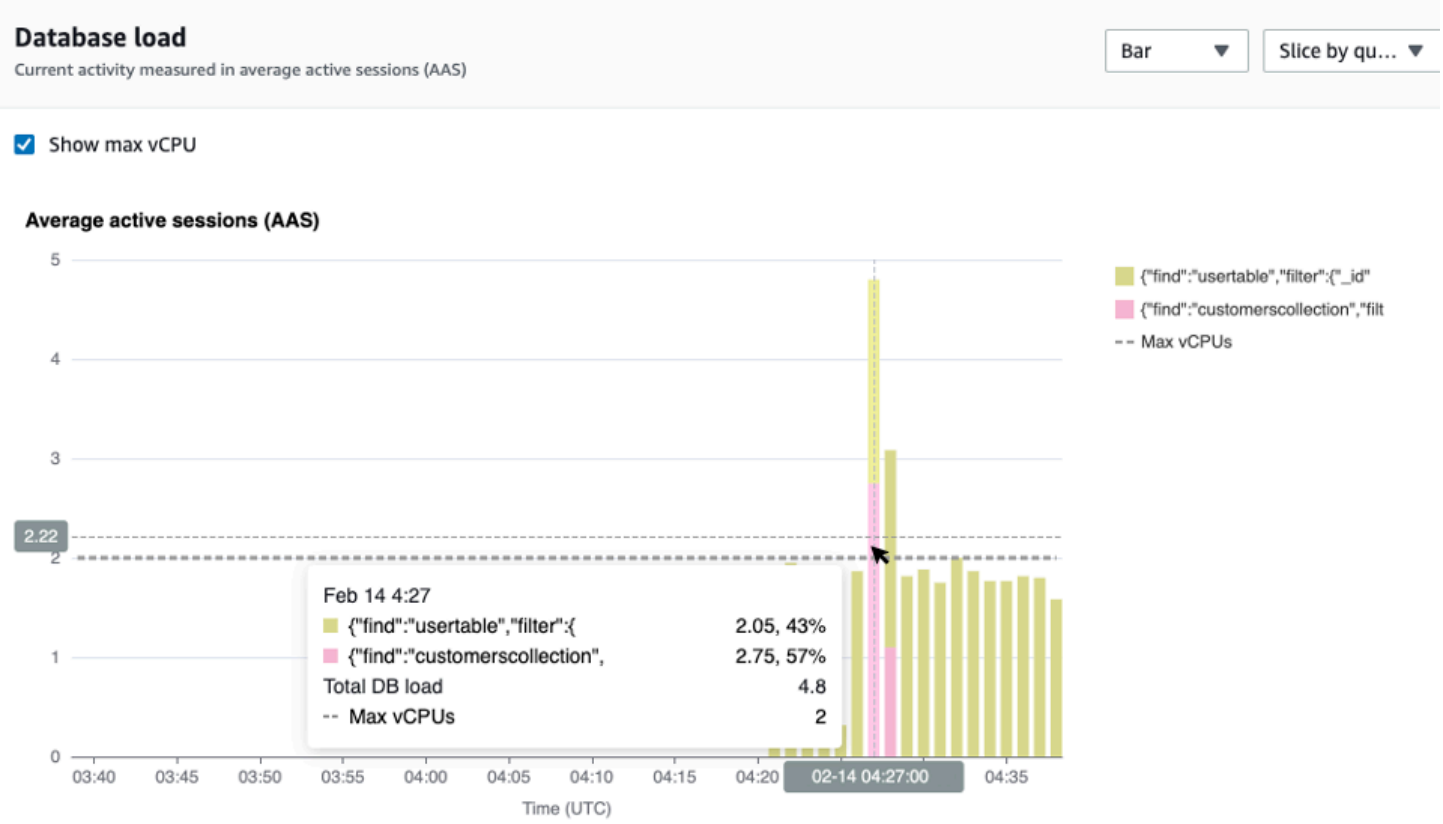
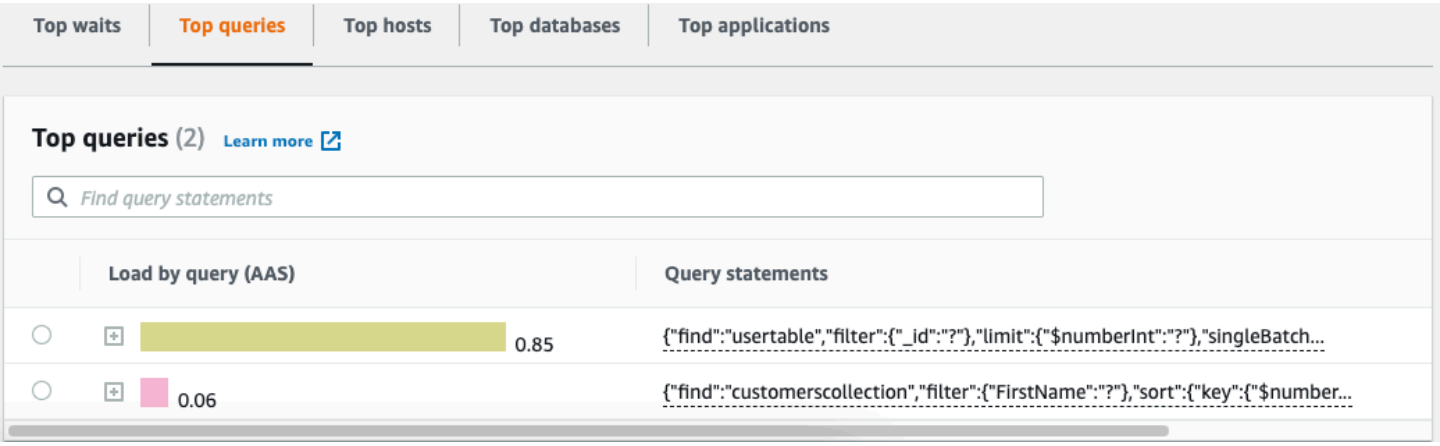


Tabla de dimensiones principales

La tabla de dimensiones principales divide la carga de base de datos por diferentes dimensiones. Una dimensión es una categoría o “dividir por” para diferentes características de la carga de base de datos. Si la dimensión es consulta, Consultas principales muestra las instrucciones de consulta que más contribuyen a la carga de bases de datos.

Elija cualquiera de las siguientes pestañas de dimensión.



En la siguiente tabla, se proporciona una breve descripción de cada pestaña.

Description (Descripción)
Esperas principales por las que la base de datos de backend está esperando.
Consultas principales de consulta que se están ejecutando
Hosts principales del host y el puerto

Dimensión (Descripción)	Descripción
Client_Id	Identificador del cliente conectado
DB	Nombre de la base de datos principal a la que está conectado el cliente.

Tes Description (Descripción)	
Aplicaciones principales	El nombre de la aplicación que está conectada a la base de datos.

Para obtener más información sobre cómo analizar las consultas mediante la pestaña Consulta principal, consulte Información general sobre la pestaña Top queries.

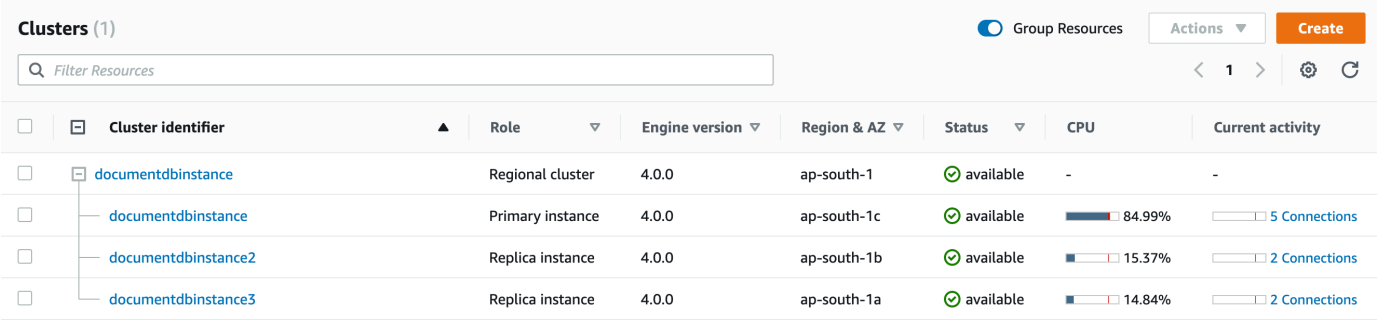
Apertura del panel de Información sobre rendimiento

Para ver el panel de Performance Insights en AWS Consola de administración, siga los pasos siguientes:

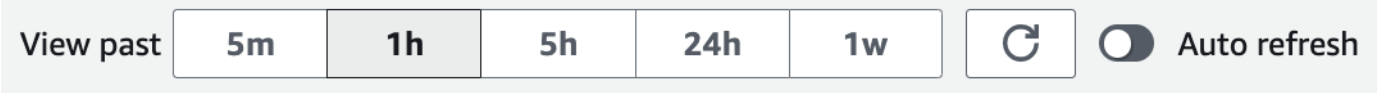
1. Abra la consola de Performance Insights en https://console.aws.amazon.com/docdb/.
2. Elija una instancia de base de datos. Se muestra el panel de Información sobre rendimiento para esa instancia de Amazon DocumentDB.

   En instancias de Amazon DocumentDB con Información sobre rendimiento habilitado, también puede acceder al panel eligiendo el elemento Sesiones en la lista de instancias de base de datos. En Actividad actual, el elemento Sesiones muestra la carga de la base de datos en el como promedio de sesiones activas en los últimos cinco minutos. La barra muestra gráficamente la carga. Cuando la barra está vacía, la instancia está inactiva. Conforme aumenta la carga, la

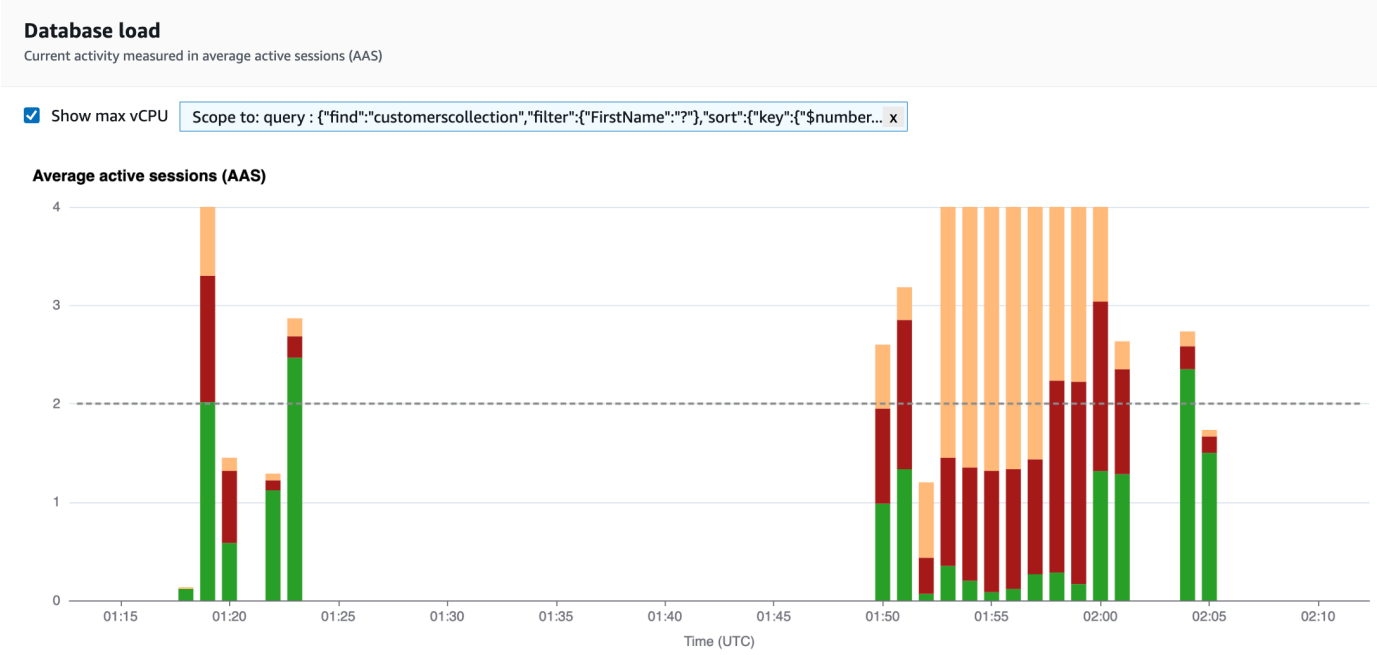
barra se va completando en azul. Cuando la carga supera el número de CPU virtuales (vCPU) en la clase de instancia, la barra cambia a rojo, lo cual indica un posible cuello de botella.



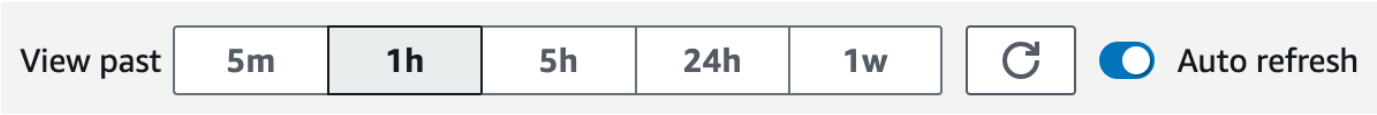
3. (Opcional) Elija un intervalo de tiempo diferente seleccionando un botón en la parte superior derecha. Por ejemplo, para cambiar el intervalo a 1 horas, seleccione 1h.



En la siguiente captura de pantalla, el intervalo de carga de la base de datos es de 1 horas.



4. Para actualizar los datos automáticamente, habilite Actualización automática.



El panel de información sobre rendimiento se actualiza automáticamente con nuevos datos. La frecuencia de actualización depende de la cantidad de datos mostrados:

- 5 minutos actualiza cada 5 segundos.
- 1 hora actualiza cada minuto.
- 5 horas actualiza cada minuto.
- 24 horas actualiza cada 5 minutos.
- 1 semana actualiza cada hora.

Análisis de carga de la base de datos mediante estados de espera

Si el gráfico Carga de base de datos indica que hay un cuello de botella, puede averiguar de dónde procede la carga. Para ello, fíjese en la tabla de elementos de carga principales situada debajo del gráfico Carga de base de datos. Elija un elemento en particular, como una consulta o un usuario, para ampliar la información de ese elemento y ver los detalles.

La carga de base de datos agrupada por esperas y principales consultas normalmente ofrece la máxima información sobre problemas de rendimiento. La carga de la base de datos agrupada por esperas indica si hay algún cuello de botella de simultaneidad o recursos en la base de datos. En este caso, la pestaña Consultas principales de la tabla de elementos de carga principales indica qué consultas están contribuyendo a esa carga.

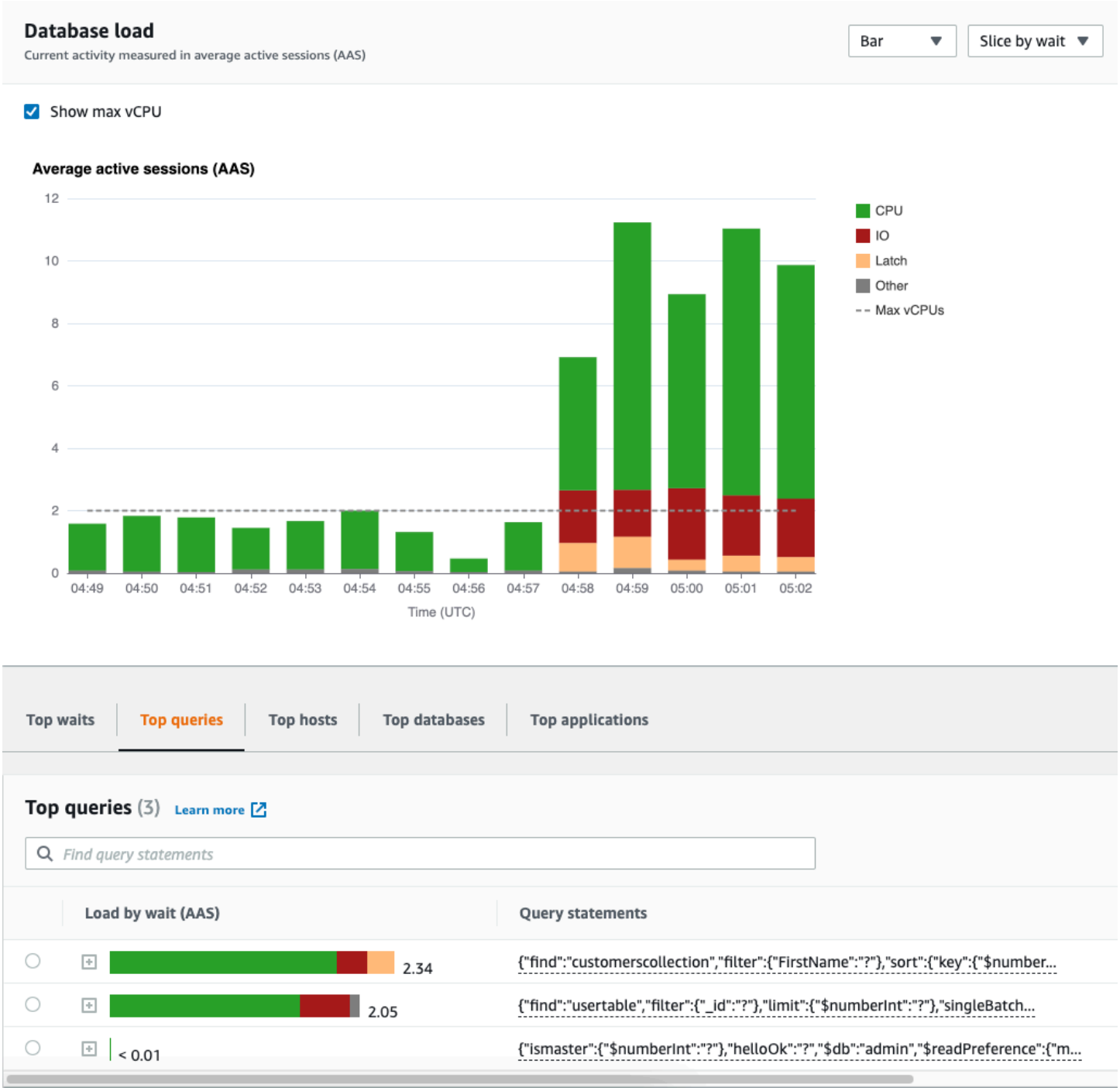
Este es el flujo de trabajo típico para diagnosticar los problemas de desempeño:

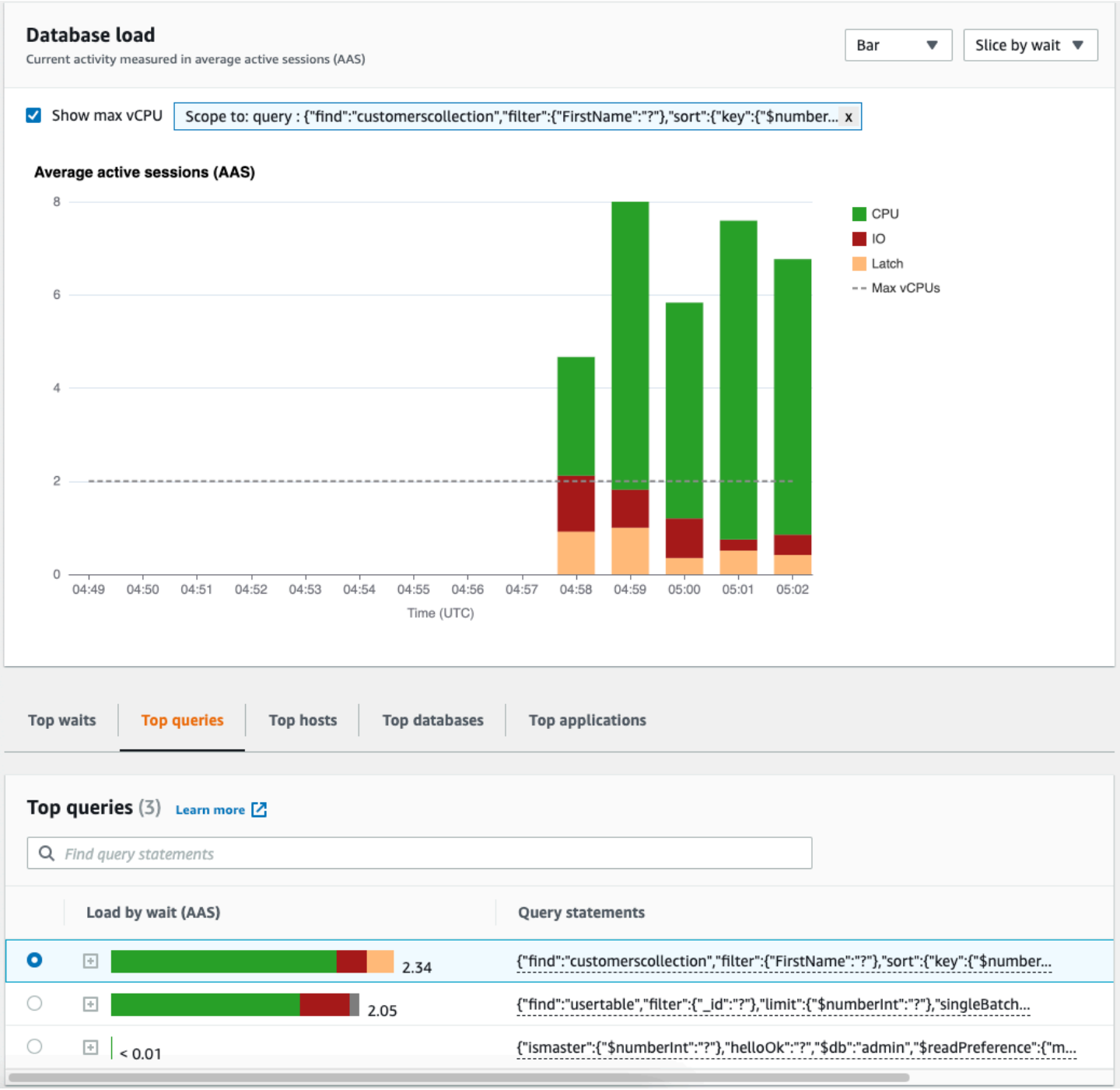
1. Revise el gráfico Carga de base de datos para ver si hay algún incidente de carga de base de datos que sobrepase la línea Máximo de CPU.
2. De ser así, fíjese en el gráfico Carga de base de datos e identifique qué estado o estados de espera son los principales responsables.
3. Para identificar las consultas de resumen que están provocando la carga, consulte qué consultas de la pestaña Top queries de la tabla de elementos de carga principales están contribuyendo más a esos estados de espera. Para identificarlas, utilice la columna Carga de base de datos por espera.
4. Elija una de estas consultas de resumen en la pestaña Top queries para ampliarla y ver las consultas secundarias que contiene.

También puede ver qué hosts o aplicaciones generan más carga seleccionando los Alojamientos principales o Aplicaciones principales, respectivamente. Los nombres de las aplicaciones se

especifican en la cadena de conexión a la instancia de Amazon DocumentDB. Unknown indica que no se especificó el campo de la aplicación.

Por ejemplo, en el panel que se muestra a continuación, la espera de la sincronización de archivos de registro se corresponde con la mayor parte de la carga de base de datos. Al seleccionar la consulta principal en Consultas principales, se analizará el diagrama de carga de la base de datos para centrarse en la mayor carga que aporta la consulta seleccionada.





Información general sobre la pestaña Top queries

De forma predeterminada, la pestaña Consultas principales muestra las consultas que más contribuyen a la carga de base de datos. Puede analizar el texto de la consulta para ajustarlas.

Temas

- Resúmenes de consultas

- Carga por esperas (AAS)
- Visualización de información detallada de la consulta
- Acceso al texto de consulta de la declaración
- Visualización y descarga del texto de consulta de la declaración

Resúmenes de consultas

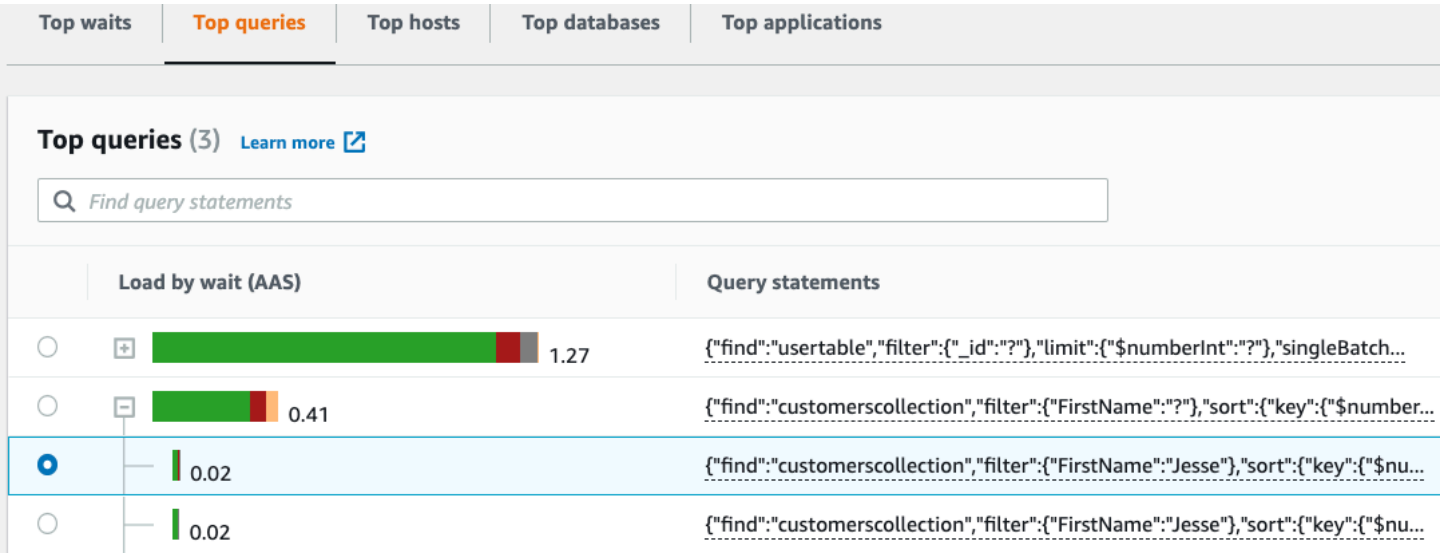
Un resumen de consultas es un compuesto de múltiples consultas reales que son similares en estructura, pero que pueden tener diferentes valores literales. El resumen reemplaza los valores codificados por un signo de interrogación. Por ejemplo, un resumen de una consulta podría verse de la siguiente manera:

```
{"find":"customerscollection","filter":{"FirstName":"?"},"sort":{"key":
{"$numberInt":"?"}},"limit":{"$numberInt":"?"}}
```

Este resumen podría incluir las siguientes consultas secundarias:

```
{"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"}}
```

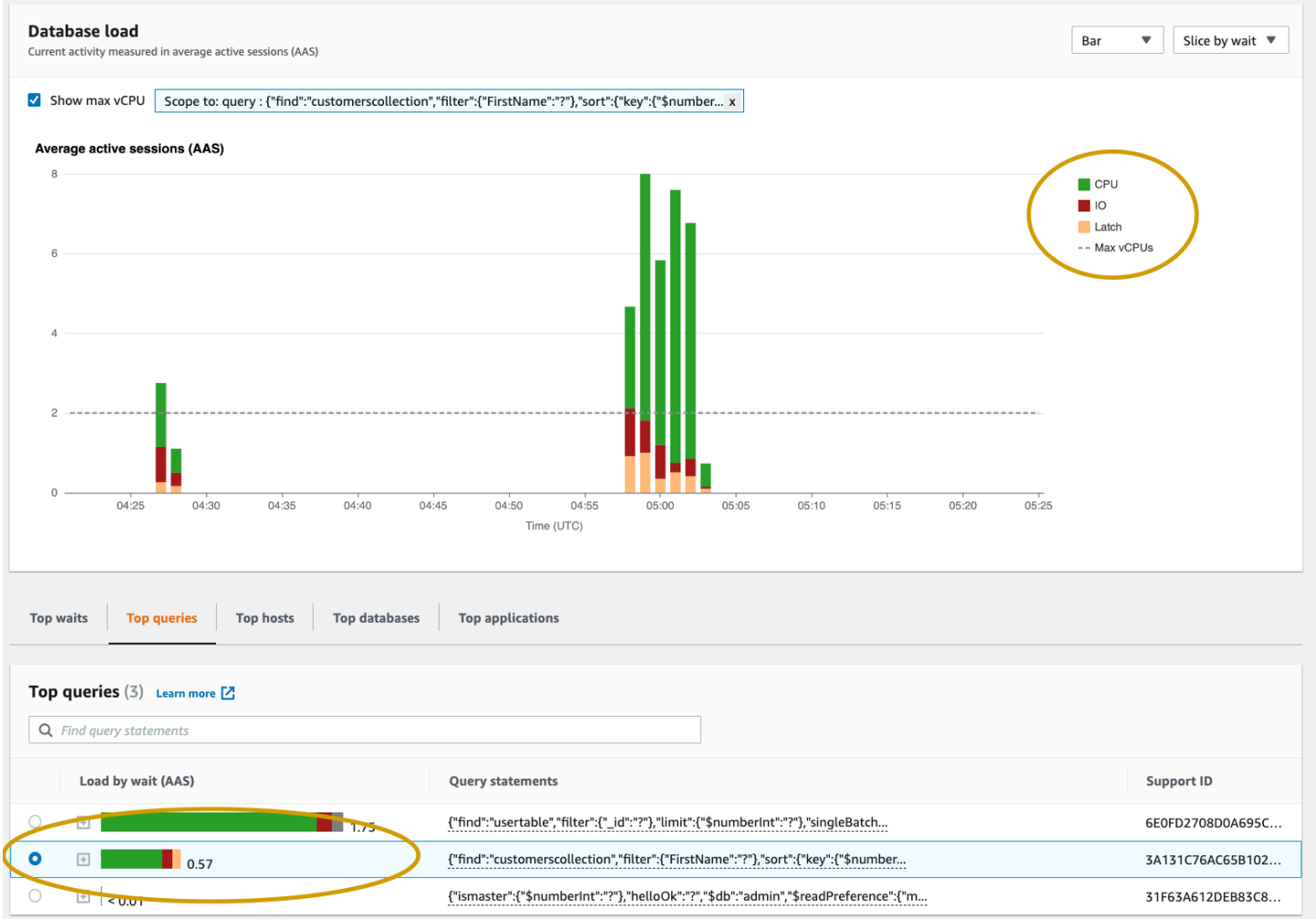
Para ver las instrucciones de consulta literales de un resumen, seleccione la consulta y, a continuación, elija el símbolo más (+). En la siguiente captura de pantalla, la consulta seleccionada es un resumen.



> **Note**
>
> Un resumen de consultas agrupa sentencias de consulta similares, pero no redacta información confidencial.

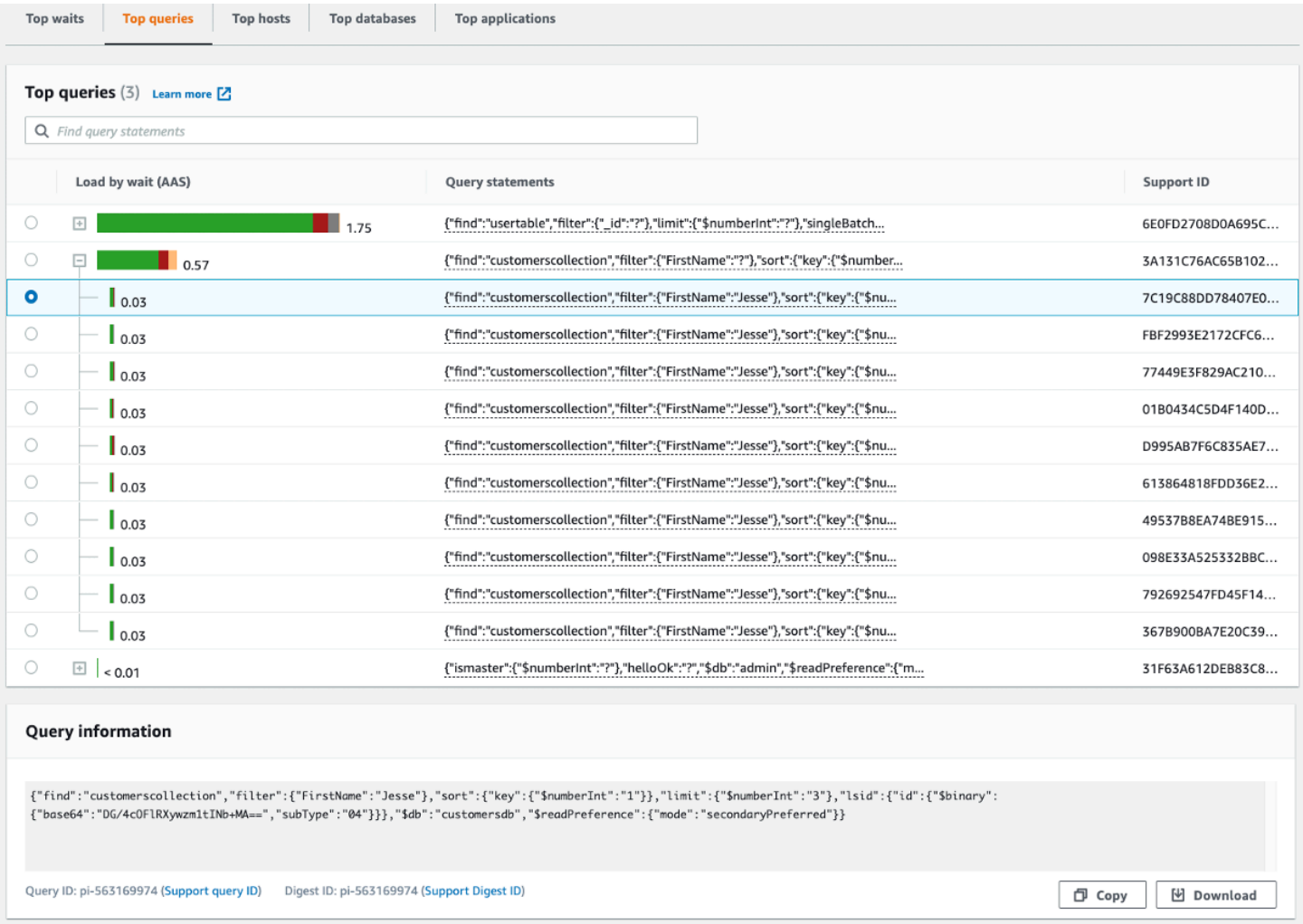
Carga por esperas (AAS)

En Consultas principales, la columna Carga por espera (AAS) ilustra el porcentaje de carga de la base de datos asociada con cada elemento de carga principal. Esta columna refleja la carga de ese elemento por cualquier agrupación que se haya seleccionado actualmente en el gráfico de carga de base de datos. Por ejemplo, es posible que pueda agrupar el gráfico DB load (Carga de base de datos) por estados de espera. En este caso, la barra DB Load by Waits (Carga de base de datos por esperas) estaría dimensionada, segmentada y dividida por colores para mostrar en qué proporción contribuye esa consulta a un estado de espera. También muestra qué estados de espera afectan a la consulta seleccionada.



Visualización de información detallada de la consulta

En la tabla Consulta principal, puede abrir un resumen de situación para consultar su información. La información aparece en el panel inferior.



Los siguientes tipos de identificadores (ID) asociados con instrucciones de consulta:

1. ID de consulta de soporte: un valor hash del ID de consulta. Este valor solo sirve para hacer referencia a un identificador de consulta cuando trabajas con el equipo de AWS soporte. AWS El equipo de soporte no tiene acceso a los identificadores ni al texto de la consulta reales.
2. Compatibilidad con ID de resumen: un valor hash del ID de resumen. Este valor solo sirve para hacer referencia a un identificador resumido cuando trabajas con AWS Support. AWS El equipo de soporte no tiene acceso a los identificadores de resumen ni al texto de la consulta reales.

Acceso al texto de consulta de la declaración

De forma predeterminada, cada fila de la tabla Consultas principales muestra 500 bytes de texto para cada instrucción. Cuando una instrucción de consulta supera los 500 bytes, puede ver más texto abriendo la instrucción en el panel de Información sobre rendimiento. En este caso, la longitud

máxima de la consulta que se muestra es de 1 KB. Si ve una instrucción SQL secundaria, también puede elegir Descargar.

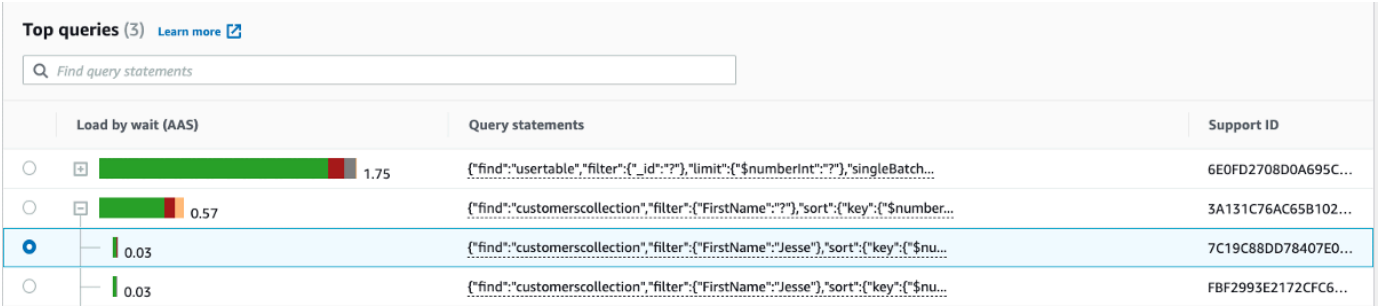
Visualización y descarga del texto de consulta de la declaración

Puede ver o descargar texto de consulta en el panel de Información sobre rendimiento.

Visualización de más texto de consulta en el panel de Información sobre rendimiento

1. Abra la consola de Amazon DocumentDB en: https://console.aws.amazon.com/docdb/
2. En el panel de navegación, seleccione Información sobre rendimiento.
3. Elija una instancia de base de datos. Se abre el panel de Información sobre rendimiento para esa instancia de base de datos.

   Las instrucciones de consulta con texto superior a 500 bytes son similares a las que se indican en la siguiente imagen.



4. Examine la sección de información de consulta para consultar más texto de consulta.



El panel de Información sobre rendimiento puede mostrar hasta 1 KB por cada instrucción de consulta completa.

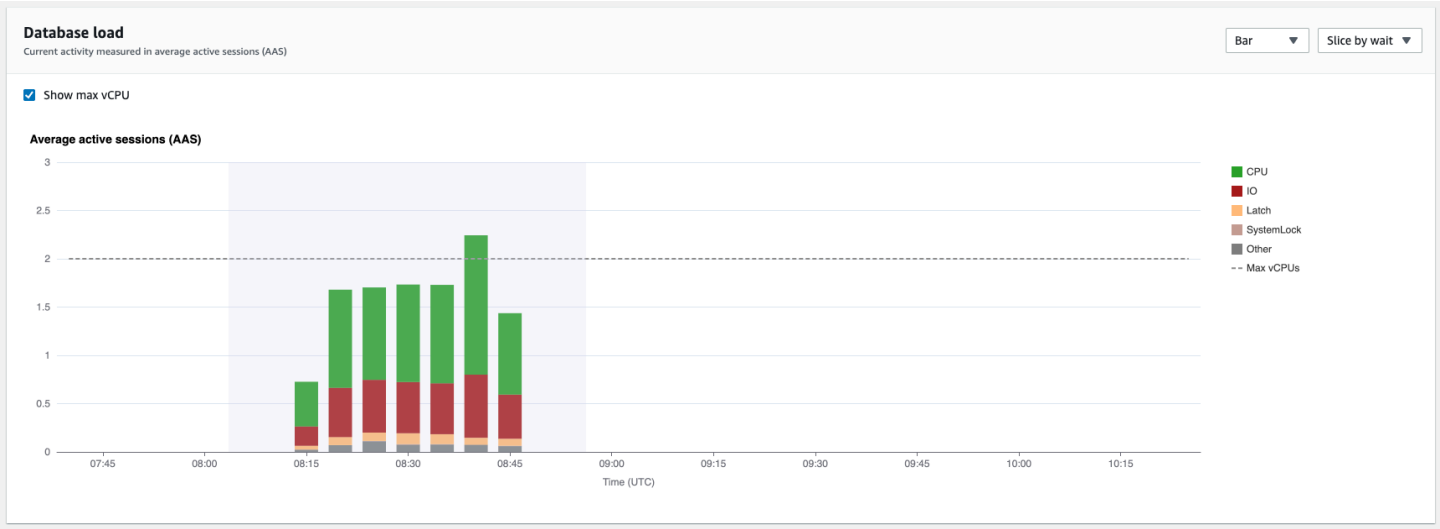
> **Note**
>
> Para copiar o descargar la instrucción de consulta, deshabilite los bloqueadores de pantallas emergentes.

Ampliación del gráfico de carga de base de datos

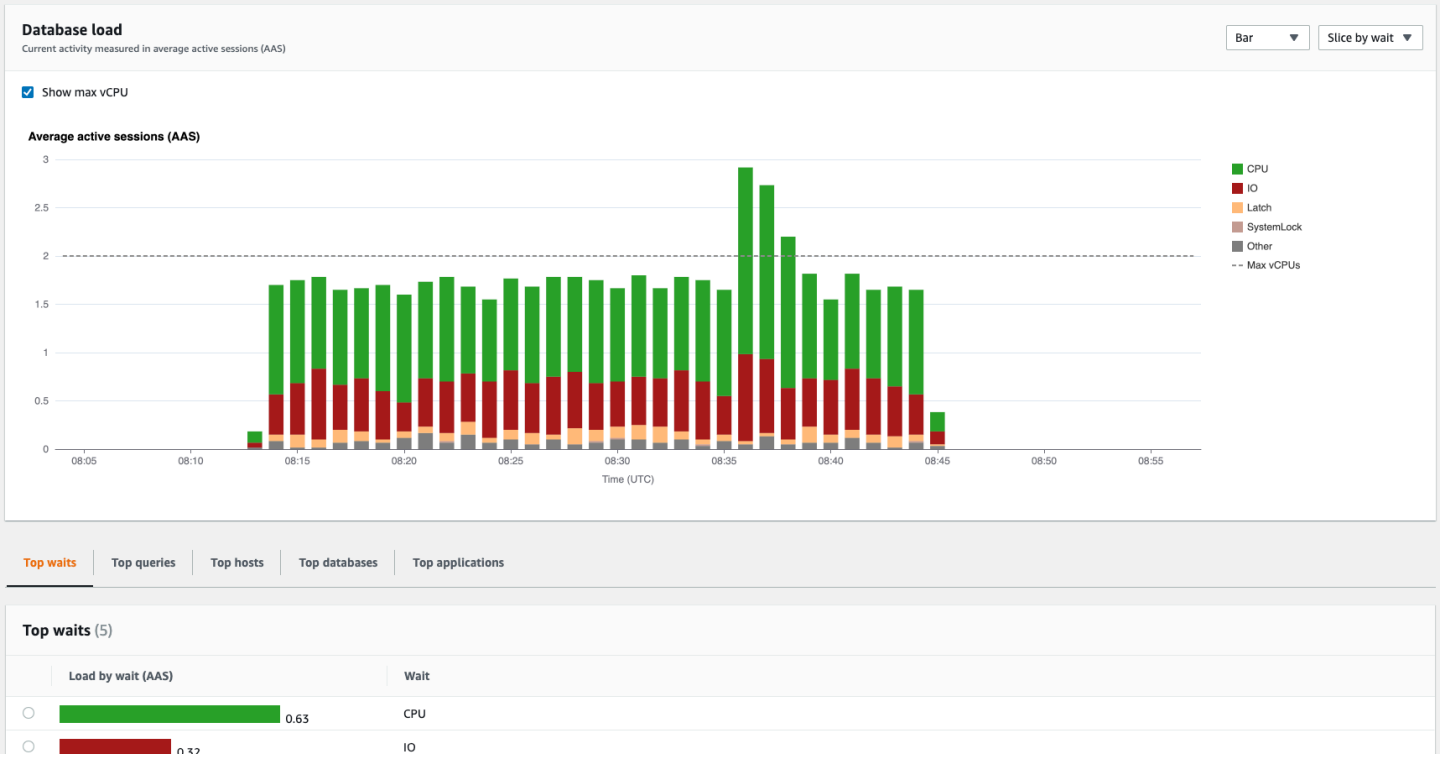
Hay otras características de la interfaz de usuario de Información sobre rendimiento que ayudan a analizar los datos de desempeño.

Click-and-Drag Ampliar

En la interfaz de Performance Insights, se puede seleccionar una pequeña parte del gráfico de carga y ampliarlo para ver los detalles.



Para ampliar una parte del gráfico de carga, elija la hora de inicio y arrastre el ratón hasta el final del período que desee. Al hacer esto, el área seleccionada queda resaltada. Cuando suelte el ratón, el gráfico de carga amplía el área seleccionada y se vuelve a calcular la tabla de Elementos principales.



Recuperación de métricas con la API de Información sobre rendimiento

Cuando se habilita Información sobre rendimiento, la API proporciona visibilidad sobre el rendimiento de la instancia. Amazon CloudWatch Logs proporciona la fuente fidedigna de las métricas de monitoreo vendidas de los AWS servicios.

Con Performance Insights se ofrece una vista propia del dominio de la carga de la base de datos entendida como el promedio de sesiones activas (AAS). Esta métrica aparece para los consumidores de API como conjunto de datos de serie temporal bidimensional. La dimensión temporal de los datos ofrece datos de carga de base de datos para cada punto temporal del intervalo de tiempo consultado. Cada punto temporal descompone la carga global en relación con las dimensiones solicitadas, tales como `Query`, `Wait-state`, `Application` o `Host`, medidas en ese punto temporal.

Información sobre rendimiento de Amazon DocumentDB monitoriza la instancia de base de datos de Amazon DocumentDB para poder analizar y solucionar los problemas de desempeño de la base de datos. Una forma de ver los datos de Información sobre rendimiento es a través de la Consola de administración de AWS. Performance Insights además ofrece una API pública, para poder consultar en sus propios datos. Puede utilizar la API para hacer lo siguiente:

- Descarga de datos en una base de datos
- Agregación de datos de Performance Insights a los paneles de monitoreo existentes

- Creación de herramientas de monitoreo

Para utilizar la API de Información sobre rendimiento, habilite Información sobre rendimiento en una de sus instancias de base de datos de Amazon DocumentDB. Para obtener información sobre la habilitación de Información sobre rendimiento, consulte Activación y desactivación de Información sobre rendimiento. Para obtener información sobre la API de Información sobre rendimiento, consulte la Referencia de la API de Información sobre rendimiento.

La API de Información sobre rendimiento proporciona las siguientes operaciones.

Acción de Performance Insights	AWS CLI comando	Description (Descripción)
`DescribeDimensionKeys`	`aws pi describe-dimension-keys`	Recupera las principales claves de dimensión N de una métrica para un periodo de tiempo específico.
`GetDimensionKeyDetails`	`aws pi get-dimension-key-details`	Recupera los atributos del grupo de dimension es especificado para una instancia de base de datos o un origen de datos. Por ejemplo, si especifica un ID de consulta y si los detalles de la dimensión están disponibl es, `GetDimensionKeyDetails` recupera el texto completo de la dimensión `db.query.statement` asociada a este ID. Esta operación resulta útil porque `GetResourceMetrics` y `DescribeDimensionKeys` no admiten la recuperac ión de texto de instrucción de consulta grande.

Acción de Performance Insights	AWS CLI comando	Description (Descripción)
GetResourceMetadata	aws pi get-resource-metadata	Recupere los metadatos de las distintas características. Por ejemplo, los metadatos podrían indicar que una característica está activada o desactivada en una instancia de base de datos específica.
GetResourceMetrics	aws pi get-resource-metrics	Recupera las métricas de Información sobre rendimiento de un conjunto de orígenes de datos, durante un periodo de tiempo. Puede proporcionar grupos de dimensiones y dimensiones específicas, y proporcionar criterios de agregación y filtrado para cada grupo.
ListAvailableResourceDimensions	aws pi list-available-resource-dimensions	Recupere las dimensiones que se pueden consultar para cada tipo de métrica especificado en una instancia especificada.
ListAvailableResourceMetrics	aws pi list-available-resource-metrics	Recupere todas las métricas disponibles de los tipos de métricas especificados que se pueden consultar de una instancia de base de datos especificada.

Temas

- AWS CLI para Performance Insights
- Recuperación de métricas de series temporales
- AWS CLI ejemplos de Performance Insights

AWS CLI para Performance Insights

Puede ver los datos de Performance Insights a través de la AWS CLI. Puede obtener ayuda sobre los comandos de la AWS CLI de Performance Insights escribiendo lo siguiente en la línea de comandos.

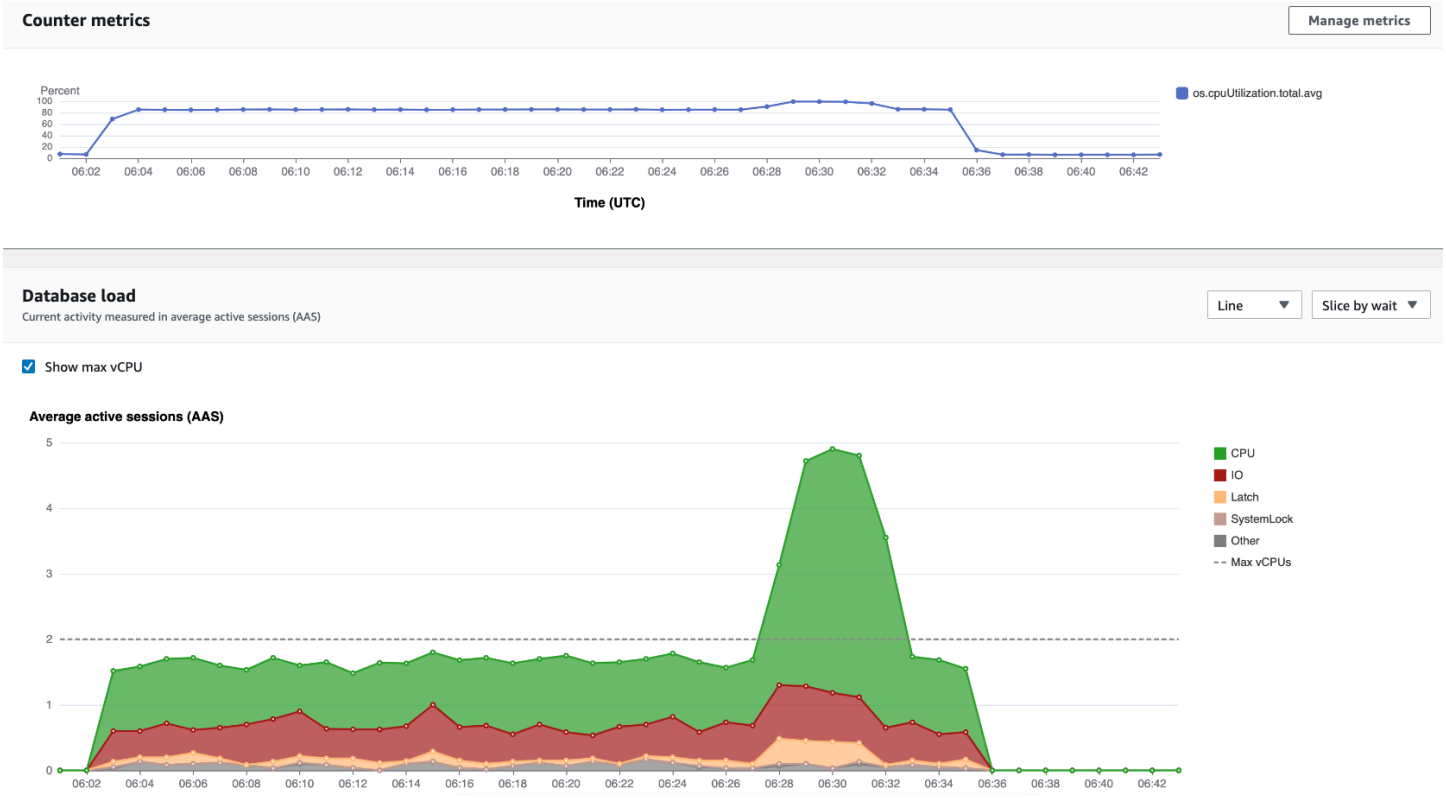
```
aws pi help
```

Si no lo tiene AWS CLI instalado, consulte Instalación de la interfaz de línea de AWS comandos en la Guía del AWS CLI usuario para obtener información sobre cómo instalarlo.

Recuperación de métricas de series temporales

La operación `GetResourceMetrics` recupera una o más métricas de series temporales a partir de los datos de Performance Insights. `GetResourceMetrics` requiere una métrica y un periodo de tiempo y devuelve una respuesta con una lista de puntos de datos.

Por ejemplo, se Consola de administración de AWS utiliza `GetResourceMetrics` para rellenar el gráfico de contrametrías y el gráfico de carga de la base de datos, como se ve en la siguiente imagen.



Todas las métricas que devuelve `GetResourceMetrics` son métricas de series temporales estándar con la excepción de `db.load`. Esta métrica se muestra en el gráfico Database Load (Carga de base de datos). La métrica `db.load` es distinta de las demás métricas de series temporales porque puede desglosarla en subcomponentes llamados dimensiones. En la imagen anterior, `db.load` está desglosado y agrupado por los estados de espera que forman el `db.load`.

> **Note**
>
> `GetResourceMetrics` también puede devolver la métrica `db.sampleload`, pero la métrica `db.load` es apropiada en la mayoría de los casos.

Para obtener información sobre las métricas de contador devueltas por `GetResourceMetrics`, consulte Métricas de contador para Información sobre rendimiento.

Para las métricas se admiten los siguientes cálculos:

- Media: el valor medio de la métrica durante un período de tiempo. Añada `.avg` al nombre de la métrica.

- Mínimo: el valor mínimo de la métrica durante un período de tiempo. Añada `.min` al nombre de la métrica.
- Máximo: el valor máximo de la métrica durante un período de tiempo. Añada `.max` al nombre de la métrica.
- Suma: la suma de los valores de la métrica durante un periodo de tiempo. Añada `.sum` al nombre de la métrica.
- Número de muestras: El número de veces que se recopiló la métrica durante un período de tiempo. Añada `.sample_count` al nombre de la métrica.

Supongamos, por ejemplo, que una métrica se recopila durante 300 segundos (5 minutos) y que la métrica se recopila una vez cada minuto. Los valores para cada minuto son 1, 2, 3, 4 y 5. En este caso, se devuelven los siguientes cálculos:

- Media: 3
- Mínimo: 1
- Máximo: 5
- Suma: 15
- Número de muestras: 5

Para obtener información sobre el uso del `get-resource-metrics` AWS CLI comando, consulte `get-resource-metrics`.

Para la opción `--metric-queries`, especifique una o más consultas para las que desea obtener resultados. Cada consulta consta de un parámetro `Metric` obligatorio y de parámetros opcionales `GroupBy` y `Filter`. A continuación, se muestra un ejemplo de una especificación de opción `--metric-queries`.

```
{
   "Metric": "string",
   "GroupBy": {
     "Group": "string",
     "Dimensions": ["string", ...],
     "Limit": integer
   },
   "Filter": {"string": "string"
     ...}
```

AWS CLI ejemplos de Performance Insights

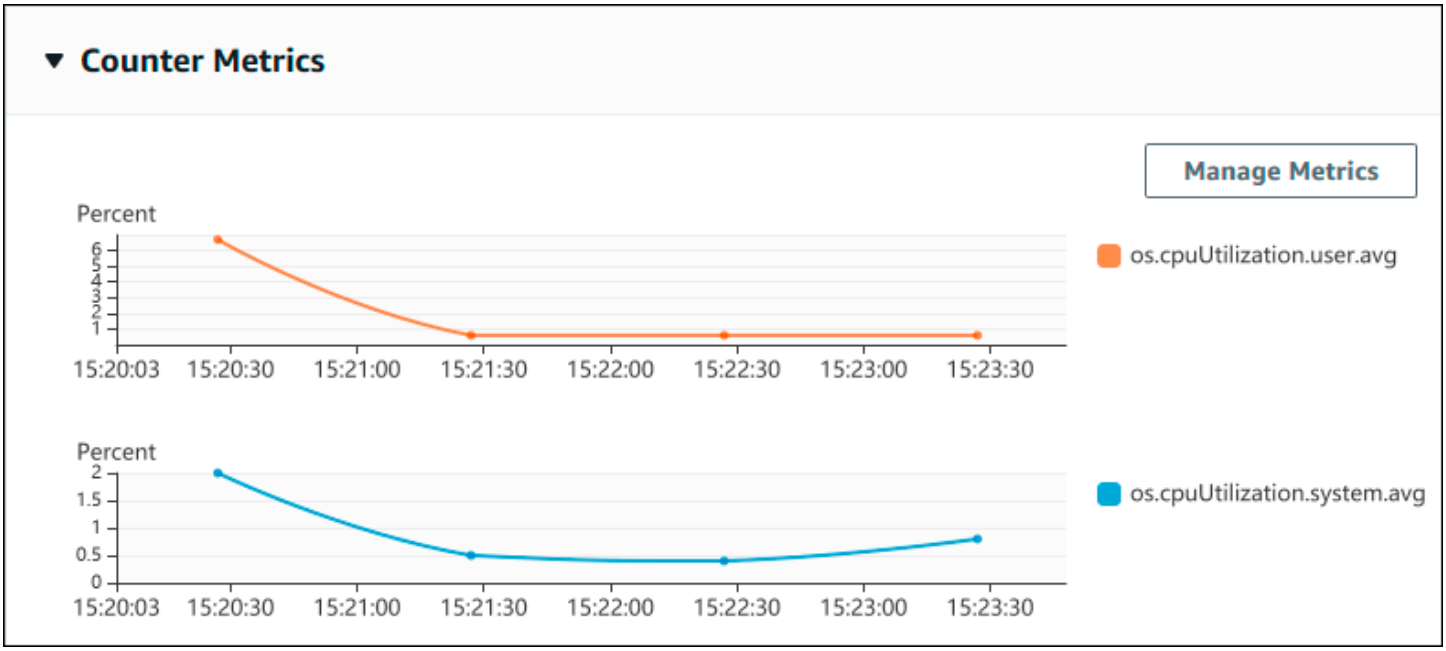
Los siguientes ejemplos muestran cómo utilizar AWS CLI for Performance Insights.

Temas

- Recuperación de métricas de contador
- Recuperación del promedio de carga de base de datos para los eventos de espera principales
- Recuperación del promedio de carga de base de datos para las instrucciones de consulta principales
- Recuperación del promedio de carga de base de datos filtrado por consulta

Recuperación de métricas de contador

La siguiente captura de pantalla muestra dos gráficos de métricas de contador en la Consola de administración de AWS.



El siguiente ejemplo muestra cómo recopilar los mismos datos que utiliza la Consola de administración de AWS para generar los dos gráficos de métricas de contador.

Para 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"}]'
```

Para 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"}]'
```

También puede hacer que un comando sea más fácil de leer especificando un archivo para la opción `--metrics-query`. El siguiente ejemplo utiliza un archivo llamado query.json para la opción. El archivo tiene el siguiente contenido.

```
[
    {
        "Metric": "os.cpuUtilization.user.avg"
    },
    {
        "Metric": "os.cpuUtilization.idle.avg"
    }
]
```

Ejecute el siguiente comando para utilizar el archivo.

Para 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
```

Para 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
```

El ejemplo anterior especifica los siguientes valores para las opciones:

- `--service-type`: DOCDB para Amazon DocumentDB
- `--identifier`: el ID de recurso para la instancia de base de datos
- `--start-time` y:`--end-time` los valores ISO 8601 `DateTime` para el periodo de consulta, con varios formatos admitidos

Consulta durante un intervalo de una hora:

- `--period-in-seconds`: `60` para una consulta por minuto
- `--metric-queries`: una matriz de dos consultas, cada una para una métrica.

  El nombre de la métrica utiliza puntos para clasificar la métrica en categorías útiles y el elemento final es una función. En el ejemplo, la función es `avg` para cada consulta. Al igual que en Amazon CloudWatch, las funciones compatibles son `min`,`max`,`total`, `avg`.

La respuesta tiene un aspecto similar a la siguiente.

```
{
    "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 respuesta tiene `Identifier`, `AlignedStartTime` y `AlignedEndTime`. Como el valor `--period-in-seconds` era `60`, los tiempos de inicio y final se han alineado con el minuto. Si el `--period-in-seconds` fuera `3600`, los tiempos de inicio y final se habrían alineado con la hora.

La `MetricList` en la respuesta tiene una serie de entradas, cada una con una entrada `Key` y una entrada `DataPoints`. Cada `DataPoint` tiene un `Timestamp` y un `Value`. Cada lista de `Datapoints` tiene 60 puntos de datos porque las consultas son datos por minuto sobre una hora, con `Timestamp1/Minute1`, `Timestamp2/Minute2` y así sucesivamente, hasta `Timestamp60/Minute60`.

Como la consulta es para dos métricas de contador distintas, hay dos elementos en la respuesta `MetricList`.

Recuperación del promedio de carga de base de datos para los eventos de espera principales

El siguiente ejemplo es la misma consulta que Consola de administración de AWS utiliza para generar un gráfico de líneas de áreas apiladas. Este ejemplo recupera el `db.load.avg` durante la última hora con la carga dividida según los siete eventos de espera principales. El comando es el mismo que el comando en Recuperación de métricas de contador. Sin embargo, el archivo query.json tiene los elementos indicados a continuación.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_state", "Limit": 7 }
    }
]
```

Ejecute el comando siguiente.

Para 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
```

Para 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
```

El ejemplo especifica la métrica de `db.load.avg` y un `GroupBy` de los siete eventos de espera principales. Para obtener más información sobre los valores válidos para este ejemplo, consulte la referencia DimensionGroup de la API Performance Insights.

La respuesta tiene un aspecto similar a la siguiente.

```
{
    "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
```

En esta respuesta, hay tres entradas en la `MetricList`. Hay una entrada para el `db.load.avg` total y tres entradas para el `db.load.avg` divididas según uno de los siete eventos de espera principales. A diferencia del primer ejemplo, como había una dimensión de agrupación, debe haber una clave para cada agrupación de la métrica. No puede haber solo una clave para cada métrica, como en el caso de uso de métrica de contador básica.

Recuperación del promedio de carga de base de datos para las instrucciones de consulta principales

El siguiente ejemplo agrupa `db.wait_state` por las 10 instrucciones de consulta principales. Hay dos grupos distintos para instrucciones de consulta:

- `db.query`: la instrucción de consulta completa, como `{"find":"customers","filter":{"FirstName":"Jesse"},"sort":{"key":{"$numberInt":"1"}}}`
- `db.query_tokenized`: la instrucción de consulta tokenizada, como `{"find":"customers","filter":{"FirstName":"?"},"sort":{"key":{"$numberInt":"?"}},"limit":{"$numberInt":"?"}}`

Al analizar el desempeño de la base de datos, puede resultar útil tener en cuenta instrucciones de consulta que solo se diferencien en sus parámetros como un elemento de lógica. Así pues, puede utilizar `db.query_tokenized` al consultar. Sin embargo, sobre todo cuando está interesado en `explain()`, a veces es más útil examinar instrucciones de consulta completas con parámetros. Existe una relación principal-secundaria entre instrucciones de consulta tokenizadas y completas, con varias instrucciones de consulta completas (secundarias) agrupadas bajo la misma instrucción de consulta tokenizada (principal).

El comando en este ejemplo es similar al comando en Recuperación del promedio de carga de base de datos para los eventos de espera principales. Sin embargo, el archivo query.json tiene los elementos indicados a continuación.

```
[
    {
        "Metric": "db.load.avg",
```

```
        "GroupBy": { "Group": "db.query_tokenized", "Limit": 10 }
    }
]
```

El siguiente ejemplo utiliza `db.query_tokenized`.

Para 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
```

Para 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
```

Este ejemplo consulta durante 1 hora, con un periodo de un minuto en segundos.

El ejemplo especifica la métrica de `db.load.avg` y un `GroupBy` de los siete eventos de espera principales. Para obtener más información sobre los valores válidos para este ejemplo, consulte DimensionGroup la referencia de la API Performance Insights.

La respuesta tiene un aspecto similar a la siguiente.

```
{
    "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
```

Esta respuesta tiene 11 entradas en la `MetricList` (1 total, 10 consultas tokenizadas principales) y cada entrada tiene 24 `DataPoints` por hora.

Para consultas tokenizadas, hay tres entradas en cada lista de dimensiones:

- `db.query_tokenized.statement`: la instrucción de consulta tokenizada.
- `db.query_tokenized.db_id` : el ID sintético que Información sobre rendimiento genera para usted. Este ejemplo devuelve el ID sintético de `pi-1064184600`.
- `db.query_tokenized.id`: el ID de la consulta dentro del panel Performance Insights.

  En Consola de administración de AWS, este identificador se denomina identificador de soporte. Se llama así porque el identificador son datos que el equipo de AWS soporte puede examinar para ayudarte a solucionar un problema con tu base de datos. AWS se toma muy en serio la seguridad y la privacidad de sus datos, y casi todos los datos se almacenan cifrados con su AWS KMS key información. Por lo tanto, nadie interno AWS puede ver estos datos. En el ejemplo anterior, tanto `tokenized.statement` como `tokenized.db_id` se almacenan cifrados. Si tienes algún problema con tu base de datos, el equipo de AWS soporte puede ayudarte haciendo referencia al identificador de soporte.

Al realizar consultas, puede ser conveniente especificar un `Group` en `GroupBy`. Sin embargo, para un control de más precisión sobre los datos que se devuelven, especifique la lista de dimensiones. Por ejemplo, si todo lo que se necesita es `db.query_tokenized.statement`, entonces se puede añadir un atributo `Dimensions` al archivo query.json.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": {
            "Group": "db.query_tokenized",
            "Dimensions":["db.query_tokenized.statement"],
            "Limit": 10
        }
    }
]
```

Recuperación del promedio de carga de base de datos filtrado por consulta

La consulta de API correspondiente en este ejemplo es similar al comando en Recuperación del promedio de carga de base de datos para las instrucciones de consulta principales. Sin embargo, el archivo query.json tiene los elementos indicados a continuación.

```
[
 {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_state", "Limit": 5  },
        "Filter": { "db.query_tokenized.id": "AKIAIOSFODNN7EXAMPLE" }
    }
]
```

En esta respuesta, todos los valores se filtran según la contribución de la consulta tokenizada AKIAIOSFODNN7EXAMPLE especificado en el archivo query.json. Las claves también podrían seguir un orden distinto de una consulta sin un filtro, porque la consulta filtrada afectaba a los cinco eventos de espera principales.

CloudWatch Métricas de Amazon para obtener información sobre el rendimiento

Performance Insights publica automáticamente las métricas en Amazon CloudWatch. Se pueden consultar los mismos datos en Performance Insights, pero tener las métricas incorporadas

CloudWatch facilita la adición de CloudWatch alarmas. También resulta fácil añadir las métricas a paneles de CloudWatch existentes.

Métrica	Descripción
DBLoad	El número de sesiones activas de Amazon DocumentDB. Normalmente, necesita los datos del número promedio de sesiones activas. En Performance Insights, estos datos se consultan como `db.load.avg` .
DBLoadCPU	El número de sesiones activas cuyo tipo de evento de espera es CPU. En Información sobre rendimiento, estos datos se consultan como `db.load.avg` , filtrados por el tipo de evento de espera `CPU`.
DBLoadNonCPU	El número de sesiones activas cuyo tipo de evento de espera no es CPU.

> **Note**
>
> Estas métricas CloudWatch solo se publican si hay carga en la instancia de base de datos.

Puede examinar estas métricas mediante la CloudWatch consola AWS CLI, la API o la CloudWatch API.

Por ejemplo, puede obtener las estadísticas para la métrica `DBLoad` ejecutando el 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
```

Este ejemplo genera un resultado similar al siguiente.

```
{
    "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"
}
```

Puede utilizar la función matemática `DB_PERF_INSIGHTS` métrica de la CloudWatch consola para consultar las métricas del contador Performance Insights de Amazon DocumentDB. La función

`DB_PERF_INSIGHTS` también incluye la métrica `DBLoad` intervalos de menos de un minuto. Puede configurar CloudWatch alarmas en estas métricas. Para obtener más información sobre cómo crear una alarma, consulte Crear una alarma en las métricas del contador de Performance Insights a partir de una AWS base de datos.

Para obtener más información al respecto CloudWatch, consulta ¿Qué es Amazon CloudWatch? en la Guía del CloudWatch usuario de Amazon.

Métricas de contador para Información sobre rendimiento

Las métricas de contador son métricas de rendimiento de sistemas operativos en el panel de control de Información sobre rendimiento. Para ayudar a identificar y analizar los problemas de rendimiento, puede correlacionar las métricas de contador con la carga de base de datos.

Contadores de sistemas operativos de Información de rendimiento

Los siguientes contadores de sistemas operativos están disponibles para Información sobre rendimiento para Amazon DocumentDB.

Contador	Tipo	Métrica
active (activa)	memory	os.memory.active
buffers	memory	os.memory.buffers
cached	memory	os.memory.cached
dirty	memory	os.memory.dirty
free	memory	os.memory.free
inactive	memory	os.memory.inactive
mapped	memory	os.memory.mapped
pageTables	memory	os.memory.pageTables
slab	memory	os.memory.slab
total	memory	os.memory.total

Contador	Tipo	Métrica
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	cargar AverageMinute	os.load AverageMinute.one
fifteen	cargar AverageMinute	os.load AverageMinute.fifteen
cinco	cargar AverageMinute	os.load AverageMinute.five
cached	swap	os.swap.cached
free	swap	os.swap.free
in	swap	os.swap.in
out	swap	os.swap.out
total	swap	os.swap.total
rx	network	os.network.rx
tx	network	os.network.tx
numVCPUs	general	os.general.numVCPUs

Uso de Amazon DocumentDB sin servidor

Amazon DocumentDB sin servidor es una configuración de escalado automático bajo demanda que automatiza los procesos de ajuste dinámico de la capacidad de la base de datos de Amazon DocumentDB en función de la demanda de las aplicaciones. Solo se le cobrará por los recursos que consuman los clústeres. Por lo tanto, DocumentDB sin servidor le permite mantenerse dentro del presupuesto y evitar pagar los recursos de computación que no utiliza.

Este tipo de automatización es especialmente valioso para entornos con cargas de trabajo muy variables e impredecibles, como bases de datos de multitenencia, bases de datos distribuidas y sistemas de desarrollo y pruebas.

Temas

- Casos de uso de DocumentDB sin servidor
- Ventajas de Amazon DocumentDB sin servidor
- Cómo funciona Amazon DocumentDB sin servidor
- Requisitos y limitaciones de DocumentDB sin servidor
- Creación de un clúster que utilice Amazon DocumentDB sin servidor
- Migración a Amazon DocumentDB sin servidor
- Administración de Amazon DocumentDB sin servidor
- Límites de instancias de Amazon DocumentDB sin servidor
- Configuración de escalado de Amazon DocumentDB sin servidor
- Supervisión de Amazon DocumentDB sin servidor

Casos de uso de DocumentDB sin servidor

Los clústeres aprovisionados de Amazon DocumentDB y DocumentDB sin servidor admiten muchos tipos de cargas de trabajo de base de datos, desde entornos de desarrollo y pruebas hasta las aplicaciones más exigentes y esenciales para la empresa que requieren una gran escala y disponibilidad. Sin embargo, DocumentDB sin servidor añade otra dimensión a las cargas de trabajo del cliente, a saber, la capacidad de admitir sitios web y aplicaciones que tienen cargas de trabajo impredecibles.

DocumentDB sin servidor es especialmente útil para los siguientes casos de uso:

- Cargas de trabajo variables: ejecuta cargas de trabajo que tienen un aumento de actividad repentino e impredecible. Un ejemplo de ello sería un sitio dedicado al tráfico, que experimenta un aumento repentino de la actividad cuando comienza a llover. Otro caso sería un sitio de comercio electrónico en el que aumenta el tráfico cuando se ofrecen ventas o promociones especiales. Con DocumentDB sin servidor, la capacidad de la base de datos se escala automáticamente para satisfacer las necesidades de los picos de carga de la aplicación y vuelve a disminuir cuando termina el aumento de actividad. Con DocumentDB sin servidor, ya no tiene que aprovisionar la capacidad para los picos ni para la carga promedio. Puede especificar un límite de capacidad superior para gestionar la peor situación. Esa capacidad no se utilizaría a menos que fuera necesario.
  - La granularidad de la reducción horizontal de DocumentDB sin servidor le permite adaptar la capacidad a las necesidades de su base de datos. En el caso de un clúster aprovisionado, el escalado vertical requiere añadir una nueva instancia. DocumentDB sin servidor puede agregar media DCU cuando solo se necesite un poco más de capacidad. Puede añadir 0,5; 1; 1,5; 2 o media DCU adicional en función de la capacidad adicional necesaria para gestionar un aumento de la carga de trabajo. Además, puede eliminar 0,5; 1; 1,5; 2 o media DCU adicional cuando la carga de trabajo disminuya y esa capacidad ya no se necesite.
- Multi-tenant aplicaciones: con DocumentDB sin servidor, no tiene que administrar de forma individual la capacidad de las bases de datos para cada aplicación de su flota. DocumentDB sin servidor administra la capacidad de las bases de datos individuales por usted.
  - Puede crear un clúster para cada inquilino. De esta forma, puede utilizar características como la clonación y la restauración de instantáneas para mejorar la alta disponibilidad y la recuperación ante desastres según corresponda para cada inquilino.
  - Cada inquilino puede tener períodos específicos de actividad e inactividad según la hora del día, la época del año, los eventos promocionales, etc. Cada clúster puede tener un amplio rango de capacidad. De esta forma, los clústeres con poca actividad incurren en los cargos mínimos de instancias. Cualquier clúster puede escalarse de forma vertical rápidamente para gestionar períodos de actividad elevada.
- Aplicaciones nuevas: si implementa una nueva aplicación y no está seguro del tamaño de instancia que necesita. Con DocumentDB sin servidor, puede configurar un clúster con una o varias instancias y hacer que la base de datos se escale automáticamente de acuerdo con los requisitos de capacidad de la aplicación.
- Mixed-use aplicaciones: supongamos que tiene una aplicación de procesamiento de transacciones en línea (OLTP), pero periódicamente experimenta picos en el tráfico de consultas. Si especifica niveles de promoción para las instancias de DocumentDB sin servidor en un clúster, puede

configurar el clúster para que las instancias de lectura se puedan escalar independientemente de la instancia de escritura para gestionar la carga adicional. Cuando el pico de uso disminuye, las instancias de lectura se reducen verticalmente de nuevo para adaptarlas a la capacidad de la instancia de escritura.

- Planificación de capacidad: supongamos que normalmente ajusta la capacidad de la base de datos o verifica la capacidad de la base de datos óptima para la carga de trabajo modificando las clases de instancias de todas las instancias de un clúster. Con DocumentDB sin servidor, puede evitar esta sobrecarga administrativa. Para determinar la capacidad mínima y máxima apropiadas, puede ejecutar la carga de trabajo y comprobar cuánto se escalan realmente las instancias.
  - Puede modificar las instancias existentes de aprovisionadas a DocumentDB sin servidor o de DocumentDB sin servidor a aprovisionadas. En estos casos, no es necesario crear un clúster nuevo ni una instancia nueva.
- Desarrollo y pruebas: además de ejecutar las aplicaciones más exigentes, también puede utilizar DocumentDB sin servidor en entornos de desarrollo y pruebas. Con DocumentDB sin servidor, puede crear instancias con una capacidad mínima en lugar de utilizar clases de instancias ampliables db.t*. Puede establecer la capacidad máxima lo suficientemente alta como para que esas instancias puedan seguir ejecutando cargas de trabajo sustanciales sin quedarse sin memoria. Cuando la base de datos no está en uso, todas las instancias se reducen verticalmente para evitar cargos innecesarios.

Uso de Amazon DocumentDB sin servidor para las cargas de trabajo aprovisionadas existentes

Supongamos que ya tiene una aplicación de DocumentDB en ejecución en un clúster aprovisionado. Para comprobar cómo funcionaría la aplicación con DocumentDB sin servidor, añada una o varias instancias de DocumentDB sin servidor al clúster existente como instancias de lectura. Puede comprobar con qué frecuencia se escalan y reducen verticalmente las instancias de lectura. Puede utilizar el mecanismo de conmutación por error de DocumentDB para promover que una instancia sin servidor de DocumentDB se encargue de escribir y comprobar cómo gestiona la carga de trabajo. read/write De esta forma, puede realizar el cambio con un tiempo de inactividad mínimo y sin cambiar el punto de conexión que utilizan las aplicaciones cliente. Para obtener más información sobre el procedimiento para convertir los clústeres existentes a DocumentDB sin servidor, consulte Migración a Amazon DocumentDB sin servidor.

Ventajas de Amazon DocumentDB sin servidor

DocumentDB sin servidor está destinado a cargas de trabajo variables o «con picos». Con cargas de trabajo tan impredecibles, podría tener dificultades para planificar cuándo cambiar la capacidad de la base de datos. También podría tener problemas para realizar cambios de capacidad con la suficiente rapidez mediante los mecanismos conocidos, como añadir instancias o cambiar clases de instancias. DocumentDB sin servidor ofrece las siguientes ventajas para facilitar estos casos de uso:

- Administración de la capacidad más sencilla que con la aprovisionada: DocumentDB sin servidor reduce el esfuerzo de planificar los tamaños de las instancias y cambiar el tamaño de las instancias a medida que cambia la carga de trabajo. También reduce el esfuerzo para mantener una capacidad uniforme para todas las instancias en un clúster.

- Escalado más rápido y sencillo durante períodos de alta actividad: DocumentDB sin servidor escala la capacidad de memoria y de computación en función de las necesidades, sin interrumpir las transacciones del cliente ni de la carga de trabajo general. La capacidad de utilizar instancias de lectura con DocumentDB sin servidor le permite aprovechar el escalado horizontal además del escalado vertical. La capacidad de usar clústeres globales de DocumentDB significa que puede distribuir la carga de trabajo de lectura sin servidor de DocumentDB entre varios. Regiones de AWS Esta capacidad es más práctica que los mecanismos de escalado de los clústeres aprovisionados.

- Cost-effective durante períodos de baja actividad: la tecnología sin servidor de DocumentDB le ayuda a evitar el sobreaprovisionamiento de sus instancias. DocumentDB sin servidor agrega recursos en incrementos granulares cuando las instancias se escalan verticalmente. Solo pagará por los recursos de la base de datos que consuma. El uso de recursos de DocumentDB sin servidor se mide por segundos. De esta forma, cuando una instancia se reduce verticalmente, la reducción del uso de recursos se registra de inmediato.

- Paridad de características con elementos aprovisionados: puede utilizar todas las funciones de DocumentDB con DocumentDB sin servidor. Por ejemplo, con DocumentDB serverless, puede usar instancias de lectura, autenticación de bases de datos AWS Identity and Access Management (IAM) y Performance Insights.

  En particular, con DocumentDB sin servidor puede aprovechar las siguientes características de los clústeres aprovisionados:

- Instancias de lectura: DocumentDB sin servidor puede aprovechar las instancias de lectura para escalar horizontalmente. Cuando un clúster contiene una o más instancias de lectura, el clúster puede conmutar por error inmediatamente en caso de que haya problemas con la instancia de escritura.
- Multi-AZ clústeres: puede distribuir las instancias sin servidor de DocumentDB de un clúster en varias zonas de disponibilidad (AZ). La configuración de un Multi-AZ clúster ayuda a garantizar la continuidad del negocio, incluso en el raro caso de que se produzcan problemas que afecten a toda una zona de disponibilidad.
- Clústeres globales: puede usar DocumentDB sin servidor en combinación con los clústeres globales de DocumentDB para crear copias adicionales de solo lectura de su clúster en otros Regiones de AWS sitios con fines de recuperación ante desastres.

Cómo funciona Amazon DocumentDB sin servidor

Temas

- Descripción general de
- Configuraciones para clústeres de Amazon DocumentDB
- Capacidad de escalado de Amazon DocumentDB sin servidor
- Escalado de Amazon DocumentDB sin servidor
- Estado inactivo (0,5 DCU)

Descripción general de

Amazon DocumentDB sin servidor es adecuado para la mayoría de las cargas de trabajo más exigentes y muy variables. Por ejemplo, el uso de la base de datos puede ser intensivo durante un corto periodo de tiempo, seguido de largos periodos de poca actividad o de ninguna actividad en absoluto. Ejemplos de ello son sitios web de venta minorista, juegos o deportes con eventos promocionales periódicos y bases de datos que generan informes cuando se necesitan. Otros son entornos de desarrollo y pruebas y nuevas aplicaciones en las que el uso podría aumentar rápidamente. Para casos como estos y muchos otros, la configuración correcta de la capacidad por anticipado no siempre es posible con el modelo aprovisionado. También puede resultar en costos más elevados si se aprovisiona en exceso y tiene capacidad que no utiliza.

En cambio, los clústeres aprovisionados de DocumentDB son adecuados para cargas de trabajo estables. Con los clústeres aprovisionados, se elige una clase de instancia que tenga una cantidad

predefinida de memoria, potencia de CPU, I/O ancho de banda, etc. Si la carga de trabajo cambia, modificará manualmente la clase de instancia del escritor y los lectores. El modelo aprovisionado funciona bien cuando se puede ajustar con anticipación la capacidad de los patrones de consumo esperados y se aceptan interrupciones breves mientras cambia la clase de instancia del escritor y los lectores del clúster.

DocumentDB sin servidor está diseñado desde cero para admitir clústeres sin servidor que son escalables instantáneamente. DocumentDB sin servidor está diseñado para proporcionar el mismo grado de seguridad y aislamiento que los lectores y escritores aprovisionados. Estos aspectos son cruciales en entornos de nube sin servidor multitenant. El mecanismo de escalado dinámico tiene muy poca sobrecarga para que pueda responder rápidamente a los cambios en la carga de trabajo de la base de datos. También es lo suficientemente potente como para satisfacer los incrementos drásticos en la demanda de procesamiento.

Con DocumentDB sin servidor, podrá crear un clúster de DocumentDB sin quedar sujeto a una capacidad de base de datos específica para cada escritor y lector. El usuario especifica el rango mínimo y máximo de capacidad. DocumentDB escala cada escritor o lector de DocumentDB sin servidor en el clúster dentro de ese rango de capacidad. Al usar un Multi-AZ clúster en el que cada escritor o lector pueda escalar de forma dinámica, puedes aprovechar el escalado dinámico y la alta disponibilidad.

DocumentDB sin servidor escala los recursos de base de datos de forma automática en función de las especificaciones de capacidad mínima y máxima. El escalado es rápido, ya que la mayoría de las operaciones de eventos de escalado mantienen al escritor o al lector en el mismo host. En los raros casos en que un escritor o lector de DocumentDB sin servidor se mueve de un host a otro, DocumentDB sin servidor administra las conexiones automáticamente. No es necesario cambiar el código de la aplicación cliente de base de datos o las cadenas de conexión de base de datos.

Con DocumentDB sin servidor, al igual que ocurre con los clústeres aprovisionados, la capacidad de almacenamiento y la capacidad informática son independientes. Cuando hablamos de capacidad y escalado de DocumentDB sin servidor, nos referimos siempre a la capacidad informática que aumenta o disminuye. Por lo tanto, el clúster puede contener muchos terabytes de datos incluso cuando la capacidad de la CPU y la memoria se reducen verticalmente a niveles bajos.

En lugar de aprovisionar y administrar los servidores de bases de datos, hay que especificar la capacidad de la base de datos. La capacidad real de cada instancia de escritura o lectura de DocumentDB sin servidor varía con el tiempo, según la carga de trabajo. Para obtener más información sobre este mecanismo, consulte Escalado de Amazon DocumentDB sin servidor.

Configuraciones para clústeres de Amazon DocumentDB

Para cada uno de sus clústeres de Amazon DocumentDB, puede elegir cualquier combinación de capacidad de DocumentDB sin servidor, capacidad aprovisionada o ambas.

Puede configurar un clúster que contenga tanto capacidad aprovisionada como capacidad de DocumentDB sin servidor. A esto se lo conoce como clúster de configuración mixta. Por ejemplo, supongamos que necesita más read/write capacidad de la que está disponible para un grabador sin servidor de DocumentDB. En este caso, puede configurar el clúster con un escritor aprovisionado muy grande. Entonces, aún puede usar DocumentDB sin servidor para los lectores. O, supongamos que la carga de trabajo de escritura del clúster varía pero la carga de trabajo de lectura es estable. En este caso, puede configurar el clúster con un escritor de DocumentDB sin servidor y uno o más lectores aprovisionados.

También puede configurar un clúster en el que DocumentDB sin servidor administre toda la capacidad. Para ello, puede crear un nuevo clúster y utilizar DocumentDB sin servidor desde el principio. O bien, puede reemplazar toda la capacidad aprovisionada de un clúster existente con DocumentDB sin servidor. Para conocer los procedimientos para crear un nuevo clúster con DocumentDB sin servidor o para cambiar un clúster existente a DocumentDB sin servidor, consulte Creación de un clúster que utilice Amazon DocumentDB sin servidor y Migración a Amazon DocumentDB sin servidor.

Si no utiliza DocumentDB sin servidor en absoluto en un clúster, todos los escritores y lectores del clúster serán aprovisionados. Este es el tipo de clúster más común con el que la mayoría de los usuarios están familiarizados. La capacidad aprovisionada es constante. Los cargos son relativamente fáciles de pronosticar. No obstante, debe predecir de antemano cuánta capacidad necesita. En algunos casos, las predicciones pueden ser inexactas o las necesidades de capacidad pueden cambiar. En estos casos, el clúster puede estar infraprovisionado (más lento de lo que se desea) o sobreaprovisionado (más caro de lo que se desea).

Capacidad de escalado de Amazon DocumentDB sin servidor

La unidad de medida de Amazon DocumentDB sin servidor es la unidad de capacidad de DocumentDB (DCU). La capacidad de escalado de DocumentDB sin servidor no está vinculada a las clases de instancias que utiliza para los clústeres aprovisionados.

Cada DCU es una combinación de aproximadamente 2 gigabytes (GiB) de memoria, la CPU correspondiente y las redes. El rango de capacidad de la base de datos se especifica mediante esta unidad de medida. Las métricas `ServerlessDatabaseCapacity` y `DCUUtilization`

CloudWatch le ayudan a determinar cuánta capacidad está utilizando realmente su base de datos y dónde se encuentra dentro del rango especificado.

Cada escritor o lector de DocumentDB sin servidor tiene una capacidad en un momento dado. La capacidad es un número de coma flotante que representa las DCU. La capacidad aumenta o disminuye cada vez que el escritor o el lector se escalan. Este valor se mide cada segundo. Para cada clúster en el que se pretende utilizar DocumentDB sin servidor, se define un rango de capacidad: los valores de capacidad mínima y máxima entre los que puede escalar cada escritor o lector de DocumentDB sin servidor. El rango de capacidad es el mismo para cada escritor o lector de DocumentDB sin servidor en un clúster. Cada escritor o lector de DocumentDB sin servidor tiene su propia capacidad, la cual corresponderá a algún punto en ese rango.

DocumentDB sin servidor es compatible con DocumentDB 5.0 y 8.0 con un rango de capacidad de 0,5 a 256 DCU.

La capacidad de DocumentDB sin servidor más baja que puede definir es de 0,5 DCU. Puede especificar un número mayor si es menor o igual que el valor de capacidad máxima admitida. Si se establece la capacidad mínima en un número pequeño, los clústeres cargados ligeramente consumen recursos informáticos mínimos. Al mismo tiempo, permanecen listos para aceptar conexiones de inmediato y escalar verticalmente cuando están ocupados.

Establezca el valor mínimo en un valor que permita a cada escritor o lector almacenar el conjunto de trabajo de la aplicación en el conjunto de búferes. De esta forma, el contenido del grupo del búfer no se desecha durante los períodos de inactividad. Para saber todo lo que debe tener en cuenta a la hora de elegir el rango de capacidad de escalado, consulte Selección del rango de capacidad de escalado para un clúster de DocumentDB sin servidor.

Según cómo se configuren los lectores en una Multi-AZ implementación, sus capacidades pueden estar vinculadas a la capacidad del escritor o de forma independiente. Para obtener información detallada sobre cómo hacerlo, consulte Visualización y modificación del nivel de promoción de los lectores sin servidor.

Supervisar DocumentDB sin servidor implica medir los valores de capacidad del escritor y los lectores del clúster a lo largo del tiempo. Si la base de datos no se reduce verticalmente a la capacidad mínima, puede realizar acciones como ajustar el mínimo y optimizar la aplicación de base de datos. Si la base de datos alcanza su capacidad máxima de forma coherente, puede realizar acciones como aumentar el máximo. También puede optimizar la aplicación de base de datos y distribuir la carga de consultas entre más lectores.

Los costos de la capacidad sin servidor de DocumentDB se miden en términos de. DCU-hours Para obtener información sobre cómo se calculan los cargos de DocumentDB sin servidor, consulte los Precios de Amazon DocumentDB. Supongamos que el número total de escritores y lectores del clúster es n. En ese caso, el clúster consume aproximadamente n x DCU mínima cuando no se está ejecutando ninguna operación de base de datos. Amazon DocumentDB podría ejecutar operaciones de supervisión o mantenimiento que provocan una pequeña cantidad de carga. Ese clúster no consume más de n x DCU máximas cuando la base de datos se ejecuta a plena capacidad.

Para obtener más información sobre cómo elegir los valores de DCU mínimos y máximos adecuados, consulte Selección del rango de capacidad de escalado para un clúster de DocumentDB sin servidor. Los valores mínimo y máximo de DCU que especifica también afectan a algunos límites de instancias de Amazon DocumentDB. Para obtener información detallada sobre la interacción entre el rango de capacidad y los límites de la instancia, consulte Límites de instancias de Amazon DocumentDB sin servidor.

Escalado de Amazon DocumentDB sin servidor

Para cada escritor o lector de DocumentDB sin servidor, Amazon DocumentDB realiza un seguimiento continuo del uso de recursos como la CPU, la memoria y la red. Estas mediciones se denominan colectivamente carga. La carga incluye las operaciones de base de datos realizadas por la aplicación. También incluye procesamiento en segundo plano para el servidor de base de datos y tareas administrativas de Amazon DocumentDB. Cuando la capacidad se ve limitada por alguno de estos factores, DocumentDB sin servidor se escala verticalmente. DocumentDB sin servidor también se escala verticalmente cuando detecta problemas de rendimiento que se pueden resolver de esta manera. Puede supervisar la utilización de los recursos y cómo afecta el escalado en DocumentDB sin servidor mediante los procedimientos en Supervisión de Amazon DocumentDB sin servidor.

La carga puede variar según el escritor o los lectores del clúster. El escritor controla las operaciones de escritura y realiza todas las modificaciones de los datos en el volumen del clúster. Los lectores pueden procesar solicitudes de solo lectura.

El escalado es la operación que aumenta o disminuye la capacidad de DocumentDB sin servidor para la base de datos. Con DocumentDB sin servidor, cada escritor y lector tiene su propio valor de capacidad actual, que se mide en DCU. DocumentDB sin servidor escala un escritor o lector hasta una capacidad superior cuando su capacidad actual es demasiado baja para gestionar la carga. Escala el escritor o el lector a una capacidad inferior cuando su capacidad en uso es superior a la necesaria.

DocumentDB sin servidor puede aumentar la capacidad de forma incremental. Cuando la demanda de carga de trabajo comienza a alcanzar la capacidad de la base de datos actual de un escritor y un lector, DocumentDB sin servidor aumenta el número de DCU de ese escritor y lector. DocumentDB sin servidor escala la capacidad en función de los incrementos necesarios para proporcionar el mejor rendimiento de los recursos consumidos. El escalado se produce en incrementos tan pequeños como 0,5 DCU. Cuanto mayor sea la capacidad en uso, mayor será el incremento de escalado y, por lo tanto, más rápido puede producirse el escalado.

Ya que el escalado en DocumentDB sin servidor es tan frecuente, granular y no disruptivo, no genera eventos discretos en la Consola de administración de AWS. En su lugar, puede medir las CloudWatch métricas de Amazon, como `serverlessDatabaseCapacity` y`DCUUtilization`, y hacer un seguimiento de sus valores mínimos, máximos y medios a lo largo del tiempo. Para obtener más información sobre la supervisión de DocumentDB sin servidor, consulte Supervisión de Amazon DocumentDB sin servidor.

El escalado vertical o la reducción vertical pueden deberse a lo siguiente:

- Utilización de la memoria
- Utilización de la CPU
- Utilización de la red
- Uso del almacenamiento

Puede supervisar estas causas del escalado up/down en las instancias sin servidor de DocumentDB. Para obtener más información, consulte Supervisión de Amazon DocumentDB sin servidor.

Puede elegir escalar el lector al mismo tiempo que el escritor asociado o independientemente del escritor. Para ello, especifique el nivel de promoción de ese lector.

- Los lectores de DocumentDB sin servidor, en niveles de promoción 0 y 1, se escalan al mismo tiempo que el escritor. Este comportamiento de escalado hace que los lectores de los niveles prioritarios 0 y 1 sean ideales para disponibilidad. Esto se debe a que siempre tienen el tamaño adecuado para asumir la carga de trabajo de escritura en caso de conmutación por error.
- Los lectores de los niveles de promoción 2 a 15 escalan independientemente del escritor. Cada lector se mantiene dentro de los valores de DCU mínimo y máximo especificados para el clúster. Cuando un lector escala independientemente de la base de datos de escritura asociada, puede pasar a quedar inactiva y reducirse verticalmente mientras el escritor continúa procesando un gran volumen de transacciones. Sigue disponible como objetivo de conmutación por error si no

hay otros lectores disponibles en niveles de promoción más bajos. No obstante, si se promueve para ser el escritor, es posible que tenga que escalar verticalmente para manejar toda la carga de trabajo del escritor.

Para obtener más información sobre cómo ver y cambiar los niveles de promoción de las instancias sin servidor, consulte Visualización y modificación del nivel de promoción de los lectores sin servidor.

El escalado de DocumentDB sin servidor puede ocurrir mientras las conexiones de la base de datos están abiertas, mientras las transacciones están en proceso, etc. DocumentDB sin servidor no espera a que llegue un punto de inactividad para empezar el escalado. El escalado no interrumpe ninguna operación de base de datos en curso.

Si la carga de trabajo requiere más capacidad de lectura de la disponible con un solo escritor y un solo lector, puede agregar lectores múltiples de DocumentDB sin servidor al clúster. Cada lector de DocumentDB sin servidor puede escalar dentro de los valores de capacidad mínimo y máximo especificados para su clúster. Puede utilizar el punto de conexión del lector del clúster para dirigir las sesiones de solo lectura a los lectores y reducir la carga en el escritor.

Que DocumentDB sin servidor realice un escalado y la rapidez con la que se producirá el escalado una vez que se inicie también dependerán de la configuración de DCU mínima y máxima para el clúster. Además, depende de si un lector está configurado para escalar junto con el escritor o independientemente de él. Para obtener información detallada sobre la configuración del escalado, consulte Configuración de escalado de Amazon DocumentDB sin servidor.

Estado inactivo (0,5 DCU)

Cuando los escritores o lectores sin servidor de Amazon DocumentDB están inactivos, las instancias sin servidor de DocumentDB admiten el escalado hasta un estado inactivo de 0,5 DCU si el clúster está configurado para ser de 0,5. MinCapacity

En estado inactivo, las instancias de DocumentDB sin servidor no tienen suficiente capacidad de procesamiento de CPU para admitir la mayoría de las cargas de trabajo de producción, pero están listas para escalarse verticalmente con rapidez para admitir una nueva carga de trabajo. En un estado no inactivo, las instancias de DocumentDB sin servidor suelen requerir, al menos, de 1,0 a 2,5 DCU. Por lo tanto, cuando las instancias sin servidor de DocumentDB pasan de un estado inactivo a un estado no inactivo, se escalan directamente a una DCU de 1,0 a 2,5 (o el valor de si es inferior). MaxCapacity

Para poder reducir el tamaño a 0,5 DCU cuando están inactivas, los límites de instancia se limitan si MinCapacity se configuran para que sean inferiores o iguales a 1,0 DCU. Para obtener más información sobre cómo la MinCapacity configuración afecta a los límites, consulte. Límites de instancias de Amazon DocumentDB sin servidor

Requisitos y limitaciones de DocumentDB sin servidor

Requisitos de características

Disponibilidad por región

Para obtener más información sobre Regiones de AWS dónde está disponible el tipo de instancia sin servidor de Amazon DocumentDB, consulte la columna Serverless de. Clases de instancias admitidas por región

El siguiente AWS CLI comando se puede utilizar para verificar las opciones exactas de instancias sin servidor de DocumentDB que se ofrecen en una región determinada:

```
aws docdb describe-orderable-db-instance-options \
        --region my_region \
        --db-instance-class db.serverless \
        --engine docdb
```

Disponibilidad de la versión del motor

DocumentDB serverless es compatible con las versiones 5.0 y 8.0 del motor Amazon DocumentDB. No está disponible en las versiones 3.6 o 4.0 del motor.

> **Note**
>
> DocumentDB sin servidor es compatible solamente con las versiones de parche 5.0.0 más recientes. Asegúrese de que su clúster esté actualizado a una versión reciente del parche del motor. Para obtener más información sobre el mantenimiento de los clústeres de Amazon DocumentDB, consulte Mantenimiento de Amazon DocumentDB.

Configuración del clúster

Antes de añadir cualquier instancia de Amazon DocumentDB sin servidor a un clúster de Amazon DocumentDB, el clúster también debe tener el conjunto de parámetros

`ServerlessV2ScalingConfiguration`. Esto define el rango de capacidad de las instancias de DocumentDB sin servidor en el clúster. Para obtener más información sobre la configuración de escalado, consulte Configuración de escalado de Amazon DocumentDB sin servidor.

Configuración del rango mínimo de capacidad de escalado para determinadas características de Amazon DocumentDB

Algunas características de Amazon DocumentDB funcionan con DocumentDB sin servidor, pero esto podría causar problemas si el rango de capacidad es inferior al necesario para los requisitos de memoria de esas características con su carga de trabajo específica. En ese caso, es posible que la base de datos no funcione tan bien como de costumbre o que se produzcan errores de falta de memoria.

Las siguientes funciones requieren una configuración de un `MinCapacity` and/or `MaxCapacity` valor superior para un funcionamiento óptimo:

- Información de rendimiento
- Clústeres globales (se aplica solo a los principales Región de AWS)
- Creación de instancias sin servidor en un clúster con un gran volumen de datos

  Esto incluye la creación de instancias sin servidor como parte de la restauración de un clúster.

Para obtener recomendaciones sobre cómo configurar el rango de capacidad adecuado (si está utilizando esta característica), consulte Selección del rango de capacidad de escalado para un clúster de DocumentDB sin servidor. Para obtener información sobre la solución de problemas si la base de datos tiene errores de falta de memoria debido a un rango de capacidad mal configurado, consulte Evitar errores de memoria insuficiente.

Creación de un clúster que utilice Amazon DocumentDB sin servidor

Creación de un clúster de Amazon DocumentDB sin servidor

Con Amazon DocumentDB sin servidor, los clústeres son intercambiables por clústeres aprovisionados. Puede tener clústeres donde algunas instancias utilicen la tecnología sin servidor y algunas instancias estén aprovisionadas.

Compruebe que la región y la versión del motor que desee sean compatibles con DocumentDB sin servidor. Consulte Requisitos y limitaciones de DocumentDB sin servidor.

Para crear un clúster de Amazon DocumentDB en el que pueda añadir instancias sin servidor, siga el mismo procedimiento que en Creación de un clúster de Amazon DocumentDB. La única diferencia es que también debe proporcionarse el argumento `ServerlessV2ScalingConfiguration`.

El argumento `ServerlessV2ScalingConfiguration` especifica el rango de capacidad de escalado de las instancias de DocumentDB sin servidor. Se compone de los valores mínimos y máximos de la unidad de capacidad de DocumentDB (DCU) que se aplican a todas las instancias de DocumentDB sin servidor en el clúster:

- El valor de `MinCapacity` especifica la capacidad de escalado mínima.
- El valor de `MaxCapacity` especifica la capacidad de escalado máxima.

Para obtener más información sobre el escalado, consulte Configuración de escalado de Amazon DocumentDB sin servidor.

Using the Consola de administración de AWS

El siguiente ejemplo Consola de administración de AWS de configuración muestra cómo crear un clúster sin servidor de DocumentDB.

1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.
2. En el panel de navegación, seleccione Clusters (Clústeres).

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

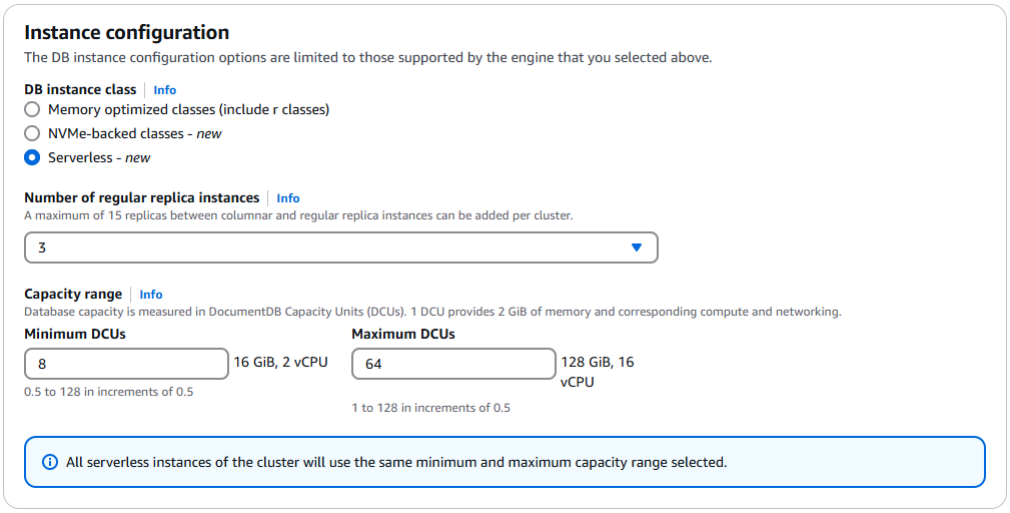
   Aparece la tabla Clústeres.
3. Seleccione Crear.

   Aparece la página Crear clúster de Amazon DocumentDB.

4. En la página Crear un clúster de Amazon DocumentDB, en la sección Tipo de clúster, elija un Instance-based clúster (es la opción predeterminada).
5. En la sección Configuración de clúster:
   a. En el campo Identificador del clúster, ingrese un nombre único, como **`myserverlesscluster`**. Tenga en cuenta que la consola cambiará todos los nombres de los clústeres a minúsculas independientemente de cómo se ingresen.
   b. En Versión del motor, elija 5.0.0 (es la opción predeterminada).
6. En la sección Configuración del almacenamiento en clúster, elija Amazon DocumentDB estándar (es la opción predeterminada).

> **Note**
>
> La otra opción de esta categoría es Amazon I/O-Optimized DocumentDB. Para obtener más información sobre las opciones, consulte Configuraciones de almacenamiento en clústeres de Amazon DocumentDB.

7. Sección de Configuración de instancias:
   a. En Clase de instancia de base de datos, elija Sin servidor.
   b. En Número de instancias de réplica normales, elija 3 (es la opción predeterminada).
   c. En la sección Rango de capacidad, deje los valores predeterminados para las DCU mínimas y las DCU máximas. Para obtener información sobre la configuración de estos parámetros, consulte Límites de instancias de Amazon DocumentDB sin servidor.



8. En la sección Conectividad, deje la configuración predeterminada de No conectarse a un recurso de computación de EC2.
9. En la sección Autenticación, ingrese un nombre de usuario para el usuario principal y, a continuación, elija Autoadministrado. Ingrese una contraseña y, a continuación, confírmela.

   Si, por el contrario, eligió Managed in AWS Secrets Manager, consulte Administración de contraseñas con Amazon DocumentDB y AWS Secrets Manager para obtener más información.
10. Deje todas las demás opciones en sus valores predeterminados y seleccione Crear clúster.

Using the AWS CLI

En el siguiente ejemplo, sustituya cada uno *user input placeholder* por su propia información o parámetros de configuración.

Para crear un clúster compatible con las instancias sin servidor de DocumentDB mediante AWS CLI, consulte. Crear un clúster con el AWS CLI

Incluya los siguientes parámetros adicionales en su comando `create-db-cluster`:

```
--serverless-v2-scaling-configuration
     MinCapacity=minimum_capacity,MaxCapacity=maximum_capacity
```

Ejemplo:

```
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
```

Añadido de una instancia de Amazon DocumentDB sin servidor

Para añadir una instancia de DocumentDB sin servidor, siga el mismo procedimiento indicado en Agregación de una instancia de Amazon DocumentDB a un clúster y asegúrese de especificar db.serverless como clase de instancia.

Agregar una instancia sin servidor mediante el Consola de administración de AWS.

Para añadir instancias de DocumentDB sin servidor mediante la consola, consulte Agregación de una instancia de Amazon DocumentDB a un clúster y seleccione la pestaña Uso de la Consola de administración de AWS.

Agregar una instancia sin servidor mediante el AWS CLI

Para agregar instancias sin servidor de DocumentDB mediante AWS CLI, consulte Agregación de una instancia de Amazon DocumentDB a un clúster y seleccione la pestaña Uso de. AWS CLI

Use el siguiente argumento de la CLI para la clase de instancia:

```
--db-instance-class db.serverless
```

Ejemplo:

```
aws docdb create-db-instance \
      --db-cluster-identifier sample-cluster \
      --db-instance-identifier sample-instance \
      --db-instance-class db.serverless \
      --engine docdb
```

Migración a Amazon DocumentDB sin servidor

Temas

- Migración de los clústeres de DocumentDB existentes a sistemas sin servidor
- Migración de MongoDB a DocumentDB sin servidor

Migración de los clústeres de DocumentDB existentes a sistemas sin servidor

Actualización de la versión del motor de un clúster

Si el clúster aprovisionado ejecuta una versión de motor inferior que no admite DocumentDB sin servidor, primero debe actualizar el clúster a una versión de motor compatible. Para obtener más información, consulte Actualización local de la versión principal Amazon DocumentDB.

Migración de un clúster aprovisionado a DocumentDB sin servidor

Para cambiar un clúster aprovisionado para que utilice DocumentDB sin servidor, siga estos pasos:

1. Compruebe si es preciso actualizar la versión del motor del clúster aprovisionado para usarla con DocumentDB sin servidor. Consulte Requisitos y limitaciones de DocumentDB sin servidor.

   > **Note**
   >
   > Si el clúster aprovisionado ejecuta una versión de motor que no está disponible para DocumentDB sin servidor, actualice la versión del motor del clúster: Consulte Actualización local de la versión principal Amazon DocumentDB.

2. Defina la configuración de escalado del clúster. Para obtener más información sobre cómo elegir la configuración de escalado, consulte Selección del rango de capacidad de escalado para un clúster de DocumentDB sin servidor. Para modificar la configuración de escalado de un clúster, consulte Visualización y modificación de la configuración del rango de capacidad de escalado de un clúster.
3. Configure cualquier otra propiedad del clúster para tener en cuenta los requisitos y las limitaciones de DocumentDB sin servidor de Requisitos y limitaciones de DocumentDB sin servidor.
4. Añada una o varias instancias de DocumentDB sin servidor al clúster. Siga el procedimiento indicado en Añadido de una instancia de Amazon DocumentDB sin servidor.

   > **Note**
   >
   > En algunos casos, es posible que ya tenga una o más instancias de lectura aprovisionadas en el clúster. De ser así, puede optar por convertir uno de los lectores en una instancia de DocumentDB sin servidor en lugar de crear una nueva instancia. Para ello, siga el procedimiento en Cambio de la clase de una instancia.

5. (Opcional) Realice una operación de conmutación por error para convertir una instancia de DocumentDB sin servidor en un clúster de escritura. Consulte Conmutación por error de Amazon DocumentDB.
6. (Opcional) Convierta las instancias de Amazon DocumentDB aprovisionadas en instancias de DocumentDB sin servidor (consulte Cambio de la clase de una instancia) o elimínelas del clúster (consulte Eliminación de una instancia de Amazon DocumentDB).

Using the Consola de administración de AWS

El siguiente ejemplo de Consola de administración de AWS configuración muestra el proceso de migración con un clúster aprovisionado de Amazon DocumentDB que ejecuta Amazon DocumentDB 5.0.0, lo que no requiere una actualización de la versión del motor para empezar a utilizar DocumentDB sin servidor. El clúster se denomina `sample` y comienza con tres instancias aprovisionadas llamadas `sample`, `sample2` y `sample3`. En este ejemplo, estas tres instancias se sustituirán por tres instancias sin servidor.

1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.
2. En el panel de navegación, seleccione Clusters (Clústeres).

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

   Aparece la tabla Clústeres.
3. En la tabla Clústeres, seleccione la casilla de verificación del clúster al que desea añadir una instancia sin servidor.
4. Elija Actions (Acciones) y después Add instances (Añadir instancias).
5. En el cuadro de diálogo Añadir instancias, seleccione Sin servidor en la sección Clase de instancias de base de datos para cada nueva instancia sin servidor que desee crear.
6. En Ajustes de capacidad sin servidor, defina la configuración de escalado en función de la descripción de la capacidad del cuadro de diálogo.
7. (Opcional) Para añadir otra instancia, elija Añadir instancia. Siga añadiendo instancias hasta alcanzar el número deseado de instancias nuevas.

   En este ejemplo, se crean tres nuevas instancias sin servidor.
8. Seleccione Crear.

   La creación de una instancia puede tardar varios minutos. Puede usar la consola o la para ver el estado de la instancia AWS CLI . Para obtener más información, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

9. Volviendo a la tabla Clústeres, seleccione las casillas de verificación de las tres instancias aprovisionadas originales.
10. Elija Acciones y, a continuación, elija Eliminar.

Durante la eliminación, se realiza automáticamente una conmutación por error para convertir a una de las instancias restantes en la instancia de escritura. Transcurridos unos minutos, se completa el proceso de eliminación. El clúster existente ahora contiene tres instancias de DocumentDB sin servidor (tal y como se define en la columna Tamaño).

Using the AWS CLI

El siguiente ejemplo de AWS CLI configuración muestra el proceso de migración con un clúster aprovisionado de Amazon DocumentDB que ejecuta Amazon DocumentDB 5.0.0, lo que no requiere actualizar la versión del motor para empezar a utilizar DocumentDB sin servidor. El clúster se denomina `sample` y comienza con tres instancias aprovisionadas llamadas `sample`, `sample2` y `sample3`. En este ejemplo, estas tres instancias se sustituirán por tres instancias sin servidor. El clúster comienza se llama `sample-cluster` y empieza con dos instancias aprovisionadas llamadas `sample-provisioned-instance-1` y `sample-provisioned-instance-2`, una instancia de escritura y una de lectura.

En el siguiente ejemplo, sustituya cada uno por su propia información o *`user input placeholder`* parámetros de configuración.

Utilice `aws docdb describe-db-clusters` para determinar el estado de un clúster. El siguiente código busca el estado del clúster `sample-cluster` y muestra los resultados en una tabla:

```
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             |
+--------------------------------+--------------------------------
+------------------+----------------+
```

Añada la configuración de escalado del clúster:

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16
```

Añada las instancias sin servidor. En este ejemplo, se añaden las nuevas instancias sin servidor denominadas `sample-serverless-instance-1` y `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
```

Introduzca lo siguiente para esperar a que las instancias sin servidor estén disponibles antes de continuar:

```
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
```

Realice una conmutación por error para que la nueva instancia de `sample-serverless-instance-1` sea el nuevo clúster de escritura:

```
aws docdb failover-db-cluster \
    --db-cluster-identifier sample-cluster \
    --target-db-instance-identifier sample-serverless-instance-1
```

La conmutación por error tarda unos segundos en completarse, tras lo cual sample-serverless-instance-1 pasa a ser el clúster de escritura. Verifíquelo con la siguiente entrada:

```
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             |
+---------------------------------+---------------------------------
+------------------+----------------+
```

Por último, elimine las instancias aprovisionadas originales:

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-provisioned-instance-1

aws docdb delete-db-instance \
    --db-instance-identifier sample-provisioned-instance-2
```

Migración de MongoDB a DocumentDB sin servidor

Puede migrar sus bases de datos de MongoDB a DocumentDB sin servidor, igual que con elementos de Amazon DocumentDB aprovisionados. Para obtener más información, consulte Migración y actualización de Amazon DocumentDB.

Administración de Amazon DocumentDB sin servidor

Visualización y modificación de la configuración del rango de capacidad de escalado de un clúster

El argumento `ServerlessV2ScalingConfiguration` especifica el rango de capacidad de escalado de las instancias de DocumentDB sin servidor. Se compone de los valores mínimos y máximos de la unidad de capacidad de DocumentDB (DCU) que se aplican a todas las instancias de DocumentDB sin servidor en el clúster.

- **`MinCapacity`**: la capacidad de escalado mínima de cualquier instancia de DocumentDB sin servidor del clúster.
- **`MaxCapacity`**: la capacidad de escalado máxima de cualquier instancia de DocumentDB sin servidor del clúster.

> **Note**
>
> Las siguientes modificaciones de la configuración de escalado requieren el reinicio de la instancia para reflejar los nuevos valores de `MinCapacity` y `MaxCapacity`:
>
> - Cualquier cambio en el valor de `MaxCapacity`
> - Cambio en el valor de `MinCapacity` a 1.0 o inferior desde un valor superior
> - Cambio en el valor de `MinCapacity` a más de 1.0 desde un valor inferior

Para obtener más información sobre la configuración del escalado y sobre cómo seleccionar los límites de capacidad de escalado adecuados, consulte Configuración de escalado de Amazon DocumentDB sin servidor.

Using the Consola de administración de AWS

El siguiente ejemplo Consola de administración de AWS de configuración muestra cómo ver y editar los ajustes de configuración de escalado de un clúster sin servidor de DocumentDB.

1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.
2. En el panel de navegación, seleccione Clusters (Clústeres).

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

   Aparece la tabla Clústeres.
3. En la tabla Clústeres, seleccione la casilla de verificación del clúster en el que desee modificar la capacidad de escalado.
4. Elija Actions (Acciones) y después Modify (Modificar).

   Aparece el cuadro de diálogo Modificar clúster.
5. Localice la sección Configuración de la capacidad sin servidor y establezca la configuración de escalado (Rango de capacidad) en función de la descripción de la capacidad del cuadro de diálogo.

   Para obtener más información acerca de los rangos de capacidad y el escalado, consulte Configuración de escalado de Amazon DocumentDB sin servidor.
6. Elija Continuar.
7. En Programación de modificaciones, elija Aplicar inmediatamente.
8. Elija Modificar clúster.
9. Una vez completada la modificación, se debe reiniciar cada instancia sin servidor. Para minimizar la falta de disponibilidad del escritor, realice la siguiente secuencia de operaciones:

   a. Reinicie cada instancia de lectura sin servidor.

      i. Seleccione la instancia de lectura, elija Acciones y, a continuación, Reiniciar.

      ii. Espere a que el estado de la instancia vuelva a Disponible.

b. Realice una conmutación por error en una instancia sin servidor reiniciada.

   i. Seleccione el clúster, elija Acciones y, a continuación, Conmutación por error.

   ii. Espere a que se complete la operación de conmutación por error.

c. Reinicie la instancia sin servidor restante.

   i. Seleccione la instancia restante, elija Acciones y, a continuación, Reiniciar.

   ii. Espere a que el estado de la instancia vuelva a Disponible.

Using the AWS CLI

El siguiente ejemplo AWS CLI de configuración muestra la configuración de escalado actual.

En el siguiente ejemplo, sustituya cada uno *user input placeholder* por su propia información o parámetros de configuración.

La configuración de escalado actual del clúster se puede ver con el `describe-db-clusters` AWS CLI comando:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[0].ServerlessV2ScalingConfiguration'
```

A continuación se muestra la salida de este comando:

```
{
    "MinCapacity": 0.5,
    "MaxCapacity": 16.0
}
```

La configuración de escalado del clúster se puede modificar mediante el comando `modify-db-cluster`:

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=32
```

Una vez completada, se debe reiniciar cada instancia sin servidor. Para minimizar la falta de disponibilidad del escritor, se puede realizar la siguiente secuencia de operaciones:

```
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
```

Visualización y modificación del nivel de promoción de los lectores sin servidor

En el caso de clústeres que incluyen múltiples instancias DocumentDB sin servidor o una combinación de instancias sin servidor y aprovisionadas, preste atención a la configuración del nivel de promoción para cada instancia sin servidor. Esta configuración controla más comportamiento para instancias sin servidor que para instancias aprovisionadas.

En el caso de las instancias aprovisionadas, la elección de nivel 0 a 15 determina únicamente el orden en que Amazon DocumentDB elige qué instancia de lectura debe promocionarse al escritor durante una operación de conmutación por error. Sin embargo, para las instancias sin servidor, el número de nivel también determina si la instancia se escala verticalmente para adaptarse a la capacidad de la instancia de escritura o si se escala independientemente en función de su propia carga de trabajo. Las instancias de lectura sin servidor de nivel 0 o 1 se mantienen con una capacidad mínima al menos igual a la de la instancia de escritura. De esta forma, quedan listas para tomar el relevo de la instancia de escritura en caso de conmutación por error. Si la instancia de escritura es una instancia aprovisionada, Amazon DocumentDB estima la capacidad equivalente de DocumentDB sin servidor. Utilice esa estimación como capacidad mínima para la instancia de lectura sin servidor.

Las instancias de lectura de DocumentDB sin servidor de los niveles 2 a 15 no tienen la misma restricción en cuanto a capacidad mínima. Cuando están inactivas, pueden reducirse verticalmente hasta el valor mínimo de la unidad de capacidad de DocumentDB (DCU) especificado en el rango de capacidad del clúster.

Using the Consola de administración de AWS

El siguiente ejemplo Consola de administración de AWS de configuración muestra cómo ver y modificar la configuración del nivel promocional de un lector de instancias sin servidor de DocumentDB.

1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.

   Los niveles de promoción de cada instancia se muestran en la columna Nivel de promoción de la Consola de administración de AWS.

2. En el panel de navegación, seleccione Clusters (Clústeres).

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

   Aparece la tabla Clústeres.

3. En la tabla Clústeres, seleccione la casilla de verificación de la instancia en la que desee modificar el nivel de promoción.
4. Elija Actions (Acciones) y después Modify (Modificar).

   Aparece el cuadro de diálogo Modificar instancia.

5. Localice la sección Conmutación por error y defina el Nivel de promoción en el nivel deseado.
6. Elija Continuar.
7. En Programación de modificaciones, elija Aplicar inmediatamente.
8. Elija Modificar instancia.

Using the AWS CLI

El siguiente ejemplo AWS CLI de configuración muestra la configuración de escalado actual.

En el siguiente ejemplo, sustituya cada uno *`user input placeholder`* por su propia información o parámetros de configuración.

Los niveles de promoción de todas las instancias de un clúster se pueden ver con el `describe-db-clusters` AWS CLI comando:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[0].DBClusterMembers' \
    --output table
```

A continuación se muestra la salida de este comando:

```
----------------------------------------------------------------------------------------------
|                                        DescribeDBClusters
                  |
+--------------------------------+---------------------------------
+------------------+----------------+
|  DBClusterParameterGroupStatus |      DBInstanceIdentifier       | IsClusterWriter
  | PromotionTier  |
+--------------------------------+---------------------------------
+------------------+----------------+
|  in-sync                       |  sample-serverless-instance-2   |  False
  |  1             |
|  in-sync                       |  sample-serverless-instance-1   |  True
  |  1             |
+--------------------------------+---------------------------------
+------------------+----------------+
```

El nivel de promoción de una instancia específica se puede modificar mediante el comando `modify-db-instance`:

```
aws docdb modify-db-instance \
    --db-instance-identifier sample-serverless-instance-2 \
    --promotion-tier 3
```

Límites de instancias de Amazon DocumentDB sin servidor

Para las instancias de DocumentDB sin servidor, los siguientes límites por instancia dependen de la capacidad de escalado actual de la instancia:

- Memoria de la instancia (GiB)
- Conexiones (todas)
- Límite del cursor
- Transacciones abiertas
- Conexiones (activas)

En las siguientes tablas, se describe cómo se reducen horizontalmente los límites por instancia de las instancias sin servidor en relación con la capacidad de escalado actual de la instancia.

Tenga en cuenta que los valores límite son diferentes en función de si la configuración de escalado de `MinCapacity` del clúster se configuró en más de dos (2). Si `MinCapacity` se establece en un valor inferior o igual a 2, los valores límite para el límite del cursor, las transacciones abiertas y las conexiones (activas) se restringen a un valor máximo inferior. (El motivo técnico del límite es permitir la reducción vertical a un estado «inactivo». Para obtener más información, consulte Estado inactivo (0,5 DCU).

db.serverless: límites de instancias si es ≤1 MinCapacity

Capacidad actual (DCU)	Memoria de la instancia (GiB)	Conexiones (todas)	Límite del cursor	Transacciones abiertas	Conexiones (activas)
0,5	1	250	6	12	39
1	2	500	12	24	79
2	4	1 000	24	48	173
4	8	2000	48	96	416
8	16	4000	96	192	1071
16	32	8000	132	264	1550

Capacidad actual (DCU)	Memoria de la instancia (GiB)	Conexiones (todas)	Límite del cursor	Transacciones abiertas	Conexiones (activas)
32	64	16 000	132	264	1550
64	128	32000	132	264	1550
128	256	60000	132	264	1550
256	512	60000	132	264	1550

límites de instancia db.serverless si es >1 MinCapacity

Capacidad actual (DCU)	Memoria de la instancia (GiB)	Conexiones (todas)	Límite del cursor	Transacciones abiertas	Conexiones (activas)
1.5	3	750	18	36	124
2	4	1 000	24	48	173
4	8	2000	48	96	416
8	16	4000	96	192	1071
16	32	8000	192	384	2709
32	64	16 000	384	768	4500
64	128	32000	768	1536	4500
128	256	60000	1536	3072	4500
256	512	60000	3072	6144	4500

Puedes supervisar y emitir alarmas sobre los límites por instancia mediante las siguientes CloudWatch métricas. Para obtener más información sobre las CloudWatch métricas de Amazon DocumentDB, consulteSupervisión de Amazon DocumentDB con CloudWatch.

Recurso	CloudWatch métrica de límite	CloudWatch métrica de uso (máximo de 1 minuto)	CloudWatch métrica de uso
Memoria de la instancia	-	-	FreeableMemory
Conexiones (todas)	DatabaseConnectionsLimit	DatabaseConnectionsMax	DatabaseConnections
Cursores	DatabaseCursorsLimit	DatabaseCursorsMax	DatabaseCursors
Transacciones	TransactionsOpenLimit	TransactionsOpenMax	TransactionsOpen

Configuración de escalado de Amazon DocumentDB sin servidor

Temas

- Selección del rango de capacidad de escalado para un clúster de DocumentDB sin servidor
- Elección de la configuración MinCapacity para un clúster de DocumentDB sin servidor
- Elección de la configuración MaxCapacity para un clúster de DocumentDB sin servidor
- Evitar errores de memoria insuficiente
- ¿Por qué no se reduce verticalmente mi instancia sin servidor?

Selección del rango de capacidad de escalado para un clúster de DocumentDB sin servidor

Antes de añadir una instancia de DocumentDB sin servidor a un clúster de Amazon DocumentDB, el clúster también debe tener el conjunto de parámetros `ServerlessV2ScalingConfiguration`.

El parámetro `ServerlessV2ScalingConfiguration` consta de dos valores que definen el rango de capacidad de escalado sin servidor de cualquier instancia sin servidor del clúster:

- **`MinCapacity`**: la capacidad de escalado mínima de cualquier instancia de DocumentDB sin servidor del clúster.

- **`MaxCapacity`**: la capacidad de escalado máxima de cualquier instancia de DocumentDB sin servidor del clúster.

Elección de la configuración **`MinCapacity`** para un clúster de DocumentDB sin servidor

Es tentador elegir siempre 0,5 para la `MinCapacity`. Este valor permite que la instancia se reduzca verticalmente a la capacidad mínima cuando esté completamente inactiva y, al mismo tiempo, permanezca activa. No obstante, en función de cómo se utilice ese clúster y de los demás ajustes que se configuren, un capacidad mínima diferente podría ser lo más efectivo. Debe tener en cuenta los factores siguientes al elegir la configuración de capacidad mínima:

- La tasa de escalado de una instancia de DocumentDB sin servidor depende de su capacidad actual. Cuanto mayor sea la capacidad actual, más rápido podrá escalar verticalmente. Si necesita que la instancia se escale verticalmente de manera rápida a una capacidad muy alta, piense en establecer la capacidad mínima en un valor en el que la tasa de escalado cumpla con sus requisitos.
- Si normalmente modifica la clase de instancia de sus instancias en previsión de una carga de trabajo especialmente alta o baja, puede aprovechar esa experiencia para hacer una estimación aproximada del rango de capacidad de DocumentDB sin servidor equivalente. Para determinar el tamaño de la memoria de una instancia de Amazon DocumentDB aprovisionada, consulte Cuotas de instancias.

  Por ejemplo, supongamos que utiliza la clase de instancia `db.r6g.xlarge` cuando el clúster tiene una carga de trabajo baja. Esta clase de instancia tiene 32 GiB de memoria. Por lo tanto, puede especificar una `MinCapacity` de 16 para configurar una instancia sin servidor que pueda reducir verticalmente hasta casi la misma capacidad. Esto se debe a que cada DCU corresponde a aproximadamente 2 GiB de memoria. Podría especificar un valor algo inferior para permitir que la instancia se reduzca verticalmente aún más en caso de que la instancia de `db.r6g.xlarge` se haya infrautilizado a veces.
- Si la aplicación funciona de la manera más eficiente cuando las instancias tienen cierta cantidad de datos en la memoria caché de búfer, piense en especificar una configuración mínima de DCU en la que la memoria sea lo suficientemente grande como para contener los datos a los que se accede con frecuencia. De lo contrario, algunos datos se expulsan de la memoria caché de búfer cuando las instancias sin servidor se reducen verticalmente a un tamaño de memoria inferior. Luego, cuando las instancias se escalan de nuevo, la información se vuelve a leer en la caché de

búfer a lo largo del tiempo. Si la cantidad necesaria I/O para devolver los datos a la caché del búfer es considerable, podría ser más eficaz elegir un valor de DCU mínimo más alto. Para obtener más información, consulte Determinación del tamaño de las instancias.

- Si las instancias de DocumentDB sin servidor se ejecutan la mayor parte del tiempo en una capacidad determinada, considere especificar una configuración de capacidad mínima inferior a esa línea de base, pero no muy inferior. Las instancias sin servidor pueden estimar de la manera más eficaz cuánto y con qué rapidez deben escalarse verticalmente cuando la capacidad actual no es drásticamente inferior a la capacidad requerida.
- Si la carga de trabajo aprovisionada tiene requisitos de memoria demasiado altos para clases de instancias pequeñas como T3 o T4g, elija una configuración de DCU mínima que proporcione memoria comparable a una instancia R5 o R6g.
- Utilice el mínimo siguiente `MinCapacity` para las funciones especificadas (sujeto a cambios):
  - Información sobre rendimiento: 2 DCU
  - Clústeres globales: 8 DCU (se aplica solo a la principal Región de AWS)
- En Amazon DocumentDB, la replicación se produce en la capa de almacenamiento, por lo que la capacidad del lector no afecta directamente a la replicación. Sin embargo, en el caso de las instancias de lectura de DocumentDB sin servidor que se escalan de forma independiente, asegúrese de que la capacidad mínima sea suficiente para gestionar las cargas de trabajo durante los períodos de escritura intensiva a fin de evitar la latencia de las consultas. Si las instancias de lectura en los niveles de promoción 2 a 15 experimentan problemas de rendimiento, considere aumentar la capacidad mínima del clúster. Para obtener más información sobre cómo cambiar si las instancias de lectura se escalan junto con el escritor o de manera independiente, consulte Visualización y modificación del nivel de promoción de los lectores sin servidor.

  Si tiene un clúster con instancias de lectura de DocumentDB sin servidor, los lectores no se escalan junto con la instancia de escritura cuando el nivel de promoción de los lectores no es 0 ni 1. En ese caso, establecer una capacidad mínima baja puede provocar un retraso excesivo de la replicación. Esto se debe a que es posible que los lectores no tengan la capacidad suficiente para aplicar cambios de escritura cuando la base de datos esté ocupada. Establezca la capacidad mínima en un valor que represente una cantidad de memoria y CPU comparable a la de la instancia de escritura.
- El tiempo que tarda una instancia de DocumentDB sin servidor en escalar desde su capacidad mínima hasta su capacidad máxima depende de la diferencia entre sus valores de DCU mínima y máxima. Cuando la capacidad actual de la instancia es grande, DocumentDB sin servidor se escala verticalmente en incrementos mayores que cuando la instancia comienza desde una

capacidad pequeña. Por lo tanto, si especifica una capacidad máxima relativamente grande y la instancia está la mayor parte del tiempo cerca de esa capacidad, considere aumentar la configuración mínima de DCU. De esta forma, una instancia inactiva podrá escalar de nuevo hasta la máxima capacidad más rápidamente.

- Algunos límites de instancia vienen determinados por la capacidad actual de la instancia sin servidor, como el límite de conexiones, el límite de cursores y el límite de transacciones abiertas. Si la capacidad actual de la instancia es pequeña, los límites también serán, en consecuencia, pequeños. Si estos límites son un problema cuando la instancia sin servidor se reduce a su valor de `MinCapacity`, considere la posibilidad de aumentar la `MinCapacity` a un valor más alto. Para obtener más información, consulte Límites de instancias de Amazon DocumentDB sin servidor.
- Además, algunos límites de instancia se limitan a un valor máximo inferior si la `MinCapacity` se establece en un valor inferior o igual a 1,0 DCU, como el límite de conexiones activas, el límite de cursores y el límite de transacciones abiertas. Si estos límites máximos no son suficientes para su carga de trabajo, utilice un `MinCapacity` valor de al menos 1,5 DCU. Para obtener más información, consulte Límites de instancias de Amazon DocumentDB sin servidor.

Para obtener instrucciones sobre cómo modificar la configuración de escalado de un clúster, consulte Administración de Amazon DocumentDB sin servidor.

Elección de la configuración **`MaxCapacity`** para un clúster de DocumentDB sin servidor

Es tentador elegir siempre un valor alto para la configuración de capacidad máxima de DocumentDB sin servidor. Una alta capacidad máxima permite que la instancia se escale verticalmente cuando ejecuta una carga de trabajo intensiva. Un valor bajo evita la posibilidad de cobros inesperados. Según cómo utilice ese clúster y los demás ajustes que configure, el valor más efectivo podría ser mayor o menor de lo que pensaba originalmente. Tenga en cuenta los factores siguientes al elegir la configuración de capacidad máxima:

- La capacidad máxima debe ser al menos tan alta como la capacidad mínima. Usted puede definir la capacidad mínima y máxima para que sean idénticas. Sin embargo, en ese caso, la capacidad nunca aumenta ni baja. Por lo tanto, utilizar valores idénticos para la capacidad mínima y máxima no es apropiado fuera de situaciones de prueba.
- La capacidad máxima debe ser 1,0 DCU como mínimo y 256 DCU como máximo.

- Supervise el escalado y el uso de recursos de sus instancias sin servidor. Si su instancia sin servidor escala con frecuencia hasta alcanzar su capacidad máxima y tiene limitaciones de recursos (por ejemplo, cuando la `DCUUtilization` métrica está en 100,0), seleccione un valor más alto. `MaxCapacity` Para obtener más información, consulte Supervisión de Amazon DocumentDB sin servidor.
- Si normalmente modifica la clase de instancia de sus instancias aprovisionadas en previsión de una carga de trabajo especialmente alta o baja, puede aprovechar esa experiencia para hacer una estimación del rango de capacidad de DocumentDB sin servidor equivalente. Para determinar el tamaño de la memoria de las instancias de Amazon DocumentDB aprovisionadas, consulte Cuotas de instancias.

  Por ejemplo, supongamos que utiliza la clase de instancia `db.r6g.4xlarge` cuando el clúster tiene una carga de trabajo elevada. Esta clase de instancia tiene 128 GiB de memoria. Por lo tanto, puede especificar una configuración de DCU máxima de 64 para configurar una instancia sin servidor que pueda escalar verticalmente hasta casi la misma capacidad. Esto se debe a que cada DCU corresponde a aproximadamente 2 GiB de memoria. Puede especificar un valor algo mayor para permitir que la instancia se escale más verticalmente en caso de que la instancia de `db.r6g.4xlarge` a veces no tenga la capacidad suficiente para gestionar la carga de trabajo de forma eficaz.
- Si tiene un límite presupuestario para usar su base de datos, elija un valor que se mantenga dentro de ese límite, incluso si todas las instancias sin servidor se ejecutan a máxima capacidad todo el tiempo. Recuerde que, cuando tenga n instancias sin servidor en el clúster, la capacidad sin servidor máxima teórica que el clúster puede consumir en cualquier momento es n veces la configuración máxima de DCU para el clúster. (La cantidad real consumida podría ser menor, por ejemplo, si algunos lectores escalan independientemente del escritor).
- Si hace uso de instancias de lectura sin servidor para descargar parte de la carga de trabajo de solo lectura de la instancia de escritura, es posible que pueda elegir una configuración de capacidad máxima inferior. Esto se hace para reflejar que cada instancia de lectura no necesita escalar tan alto como si el clúster contuviera solo una instancia.
- Supongamos que desea protegerse contra el uso excesivo por parámetros de base de datos mal configurados o consultas ineficientes en la aplicación. En ese caso, puede evitar un uso excesivo por error eligiendo un ajuste de capacidad máxima inferior al más alto absoluto que podría establecer.
- Si los picos por actividad real del usuario son poco frecuentes, pero sí ocurren, puede tener en cuenta esas ocasiones al elegir la configuración de capacidad máxima. Si la prioridad es que la aplicación siga funcionando con un rendimiento y escalabilidad completos, puede especificar una

configuración de capacidad máxima superior a la observada en el uso normal. Si está bien que la aplicación se ejecute con un rendimiento reducido durante picos de actividad muy extremos, puede elegir una configuración de capacidad máxima ligeramente inferior. Asegúrese de elegir una configuración que aún tenga suficiente memoria y recursos de CPU para mantener la aplicación en ejecución.

- Si habilita opciones de configuración en el clúster que aumenten el uso de memoria para cada instancia, tenga en cuenta esa memoria al decidir el valor máximo de DCU. Estas configuraciones incluyen las de Performance Insights y las de los clústeres globales. Asegúrese de que el valor máximo de la DCU permite que las instancias sin servidor se escalen verticalmente lo suficiente como para gestionar la carga de trabajo cuando se usen esas características. Para obtener información sobre cómo solucionar problemas causados por la combinación de una configuración de DCU máxima baja y características de Amazon DocumentDB que impongan sobrecarga de memoria, consulte Evitar errores de memoria insuficiente (a continuación).
- Utilice el mínimo siguiente `MaxCapacity` para las funciones especificadas (sujeto a cambios):
  - Creación de instancias sin servidor en un clúster con un gran volumen de datos: 2 DCU (esto incluye la creación de instancias sin servidor como parte de la restauración de un clúster).
- Algunos límites de instancia vienen determinados por la capacidad actual de la instancia, como el límite de conexiones, el límite de cursores y el límite de transacciones abiertas. A la hora de elegir el `MaxCapacity` valor de tu carga de trabajo, asegúrate de tener en cuenta estos límites de instancias para evitar que uno de estos límites suponga un obstáculo. Para obtener más información, consulte Límites de instancias de Amazon DocumentDB sin servidor.

Para obtener instrucciones sobre cómo modificar la configuración de escalado de un clúster, consulte Administración de Amazon DocumentDB sin servidor.

Evitar errores de memoria insuficiente

Si una de sus instancias de DocumentDB sin servidor alcanza constantemente el límite de su capacidad máxima, Amazon DocumentDB indica esta condición y establece la instancia en un estado de incompatible-parameters. Si bien la instancia tiene el estado incompatible-parameters, algunas operaciones quedan bloqueadas. Por ejemplo, no se puede actualizar la versión del motor. Para obtener más información sobre el estado de una instancia de Amazon DocumentDB, consulte. Supervisión del estado de un clúster de Amazon DocumentDB

Normalmente, la instancia entra en este estado cuando se reinicia con frecuencia debido a errores de falta de memoria. Amazon DocumentDB registra un evento cuando se produce este tipo de reinicio.

Para ver los eventos de los recursos, consulte Visualización de eventos de Amazon DocumentDB. Los casos de uso de memoria inusualmente elevados pueden producirse debido a una sobrecarga en el ajuste de configuraciones, como Información sobre rendimiento. También puede provenir de una carga de trabajo pesada en la instancia o de la administración de metadatos asociados a un gran número de objetos de esquema.

Si la presión de memoria disminuye para que la instancia no alcance su capacidad máxima muy a menudo, Amazon DocumentDB cambia automáticamente el estado de la instancia a disponible.

Para recuperarse de esta situación, puede llevar a cabo algunas o todas las acciones siguientes:

- Aumente el límite inferior de capacidad para instancias sin servidor. Para ello, cambie el valor mínimo de la unidad de capacidad de DocumentDB (DCU) para el clúster. Al hacerlo, se evitan problemas en los que la capacidad de una base de datos inactiva se escala a menos memoria de la necesaria para las características activadas en el clúster. Después de cambiar la configuración de DCU del clúster, reinicie la instancia sin servidor. Cuando lo hace, se evalúa si Amazon DocumentDB puede restablecer el estado de nuevo a disponible.
- Aumente el límite superior de capacidad para instancias sin servidor. Para ello, cambie el valor máximo de DCU para el clúster. Al hacerlo, se evitan problemas en los que una base de datos ocupada no puede escalar verticalmente a una capacidad con suficiente memoria para las funciones activadas en el clúster y la carga de trabajo de la base de datos. Después de cambiar la configuración de DCU del clúster, reinicie la instancia sin servidor. Cuando lo hace, se evalúa si Amazon DocumentDB puede restablecer el estado de nuevo a disponible.
- Desactivar los ajustes de configuración que requieren sobrecarga de memoria. Por ejemplo, suponga que tiene activada una característica como Información sobre rendimiento, pero no la usa. Si es así, puede desactivarla. O bien, puede ajustar los valores de capacidad mínima y máxima del clúster para tener en cuenta la memoria utilizada por esos tipos de características. Para ver instrucciones sobre cómo elegir la configuración de capacidad mínima y máxima, consulte Selección del rango de capacidad de escalado para un clúster de DocumentDB sin servidor.
- Reduzca la carga de trabajo de la instancia. Por ejemplo, puede añadir instancias de lectura al clúster para distribuir la carga de las consultas de solo lectura entre más instancias.

¿Por qué no se reduce verticalmente mi instancia sin servidor?

En algunos casos, DocumentDB sin servidor no se reduce verticalmente a la capacidad mínima, incluso sin carga en la base de datos. Esto podría deberse por alguna de las siguientes razones:

- Información sobre rendimiento puede aumentar el uso de los recursos e impedir que la base de datos se reduzca verticalmente a su capacidad mínima. Estas son algunas de ellas:
- Si una instancia de lectura no se reduce verticalmente al mínimo y permanece en la misma capacidad o más que la instancia de escritura, compruebe el nivel de prioridad de la instancia de lectura. Las instancias de lectura de DocumentDB sin servidor de nivel 0 o 1 se mantienen con una capacidad mínima al menos igual a la de la instancia de escritura. Cambie el nivel de prioridad del lector a 2 o más para que se escale y se reduzca verticalmente independientemente del escritor. Para obtener más información, consulte Escalado de Amazon DocumentDB sin servidor.
- Las cargas de trabajo pesadas de las bases de datos pueden aumentar el uso de recursos.
- Los grandes volúmenes de bases de datos pueden aumentar el uso de recursos. Amazon DocumentDB utiliza recursos de memoria y CPU para la administración de clústeres. Amazon DocumentDB requiere más CPU y memoria para administrar los clústeres con volúmenes de bases de datos más grandes. Si la capacidad mínima del clúster es inferior al mínimo requerido para la administración del clúster, el clúster no se reducirá verticalmente a la capacidad mínima.
- La actividad de mantenimiento en segundo plano puede aumentar periódicamente el uso de recursos.

Si la base de datos aún no se reduce a la capacidad mínima configurada, detenga y reinicie la base de datos para recuperar cualquier fragmento de memoria que pueda haberse acumulado con el tiempo.

> **Note**
>
> Detener e iniciar una base de datos provoca un tiempo de inactividad. Hágalo solo cuando sea necesario.

Supervisión de Amazon DocumentDB sin servidor

Para obtener más información sobre la supervisión en Amazon DocumentDB, consulte Supervisión y registro en Amazon DocumentDB.

Temas

- Memoria insuficiente: estado de parámetros incompatibles
- CloudWatch Métricas de Amazon para DocumentDB sin servidor
- Supervisión del rendimiento de DocumentDB sin servidor con Información sobre rendimiento

Memoria insuficiente: estado de parámetros incompatibles

Si una de sus instancias sin servidor alcanza constantemente el límite de su capacidad máxima, Amazon DocumentDB indica esta condición y establece la instancia en un estado de incompatible-parameters. Para obtener más información, consulte Evitar errores de memoria insuficiente.

CloudWatch Métricas de Amazon para DocumentDB sin servidor

Para obtener más información sobre el uso CloudWatch con Amazon DocumentDB, consulte. Supervisión de Amazon DocumentDB con CloudWatch

Puede ver sus instancias sin servidor CloudWatch para supervisar la capacidad consumida por cada instancia con la `ServerlessDatabaseCapacity` métrica. También puede supervisar todas las CloudWatch métricas estándar de DocumentDB, como `DatabaseConnections` las consultas. Para ver la lista completa de CloudWatch métricas que puede supervisar para Amazon DocumentDB, consulte. Supervisión de Amazon DocumentDB con CloudWatch Es importante supervisar las siguientes métricas a CloudWatch nivel de instancia para que pueda comprender cómo aumentan y disminuyen las instancias sin servidor de DocumentDB. Todas estas métricas se calculan cada segundo. De esta forma, puede supervisar el estado en uso de sus instancias sin servidor. Puede configurar alarmas para notificarle si alguna de las instancias sin servidor se acerca a un umbral de métricas relacionadas con la capacidad. Puede determinar si los ajustes de capacidad mínima y máxima son apropiados o si necesita ajustarlos. Puede determinar dónde debe centrar sus esfuerzos para optimizar la eficiencia de la base de datos.

- **`ServerlessDatabaseCapacity`**: como métrica en el nivel de instancia, informa del número de DCU representadas por la capacidad de instancia actual. Como métrica en el nivel de clúster, representa el promedio de los valores de `ServerlessDatabaseCapacity` de todas las instancias de DocumentDB sin servidor del clúster.
- **`DCUUtilization.`**: esta métrica es nueva en DocumentDB sin servidor. Este valor se representa como un porcentaje. Se calcula como el valor de la métrica `ServerlessDatabaseCapacity` dividida por el valor máximo de DCU del clúster. Tenga en cuenta las siguientes pautas para interpretar esta métrica y tomar medidas:
  - Si esta métrica se aproxima a un valor de `100.0`, la instancia se ha escalado verticalmente al punto máximo posible. Considere aumentar la configuración máxima de DCU para el clúster. De este modo, las instancias de escritura y lectura podrán escalarse a una mayor capacidad.
  - Supongamos que una carga de trabajo de solo lectura hace que una instancia de lectura se aproxime a un valor `DCUUtilization` de `100.0`, mientras que la instancia de escritura no está

cerca de su capacidad máxima. En ese caso, considere añadir instancias de lectura adicionales al clúster. De esta forma, puede distribuir la parte de solo lectura de la carga de trabajo en más instancias, lo que reduce la carga en cada instancia de lectura.

- Supongamos que está ejecutando una aplicación de producción, donde el rendimiento y la escalabilidad son las principales consideraciones. En este caso, puede establecer el valor máximo de DCU del clúster en un número elevado. Su objetivo es que la métrica `DCUUtilization` quede siempre por debajo de 100,0. Con un valor de DCU máximo alto, puede estar seguro de tener suficiente espacio en caso de que haya picos inesperados en la actividad de la base de datos. Solo se le cobrará por la capacidad de base de datos que se consuma realmente.

- **`CPUUtilization`**: esta métrica se interpreta de forma diferente en DocumentDB sin servidor que en las instancias aprovisionadas. Para DocumentDB sin servidor, este valor es un porcentaje que se calcula como la cantidad de CPU que se utiliza actualmente dividida por la capacidad de CPU disponible bajo el valor máximo de DCU del clúster. Amazon DocumentDB supervisa este valor automáticamente y escala verticalmente su instancia sin servidor cuando esta utiliza sistemáticamente una elevada proporción de su capacidad de CPU.

  Si esta métrica se aproxima a un valor de `100.0`, la instancia ha alcanzado su capacidad máxima de CPU. Considere aumentar la configuración máxima de DCU para el clúster. Si esta métrica se aproxima a un valor de `100.0` en una instancia de lectura, considere añadir instancias de lectura adicionales al clúster. De esta forma, puede distribuir la parte de solo lectura de la carga de trabajo distribuida en más instancias, y así reducir la carga en cada instancia de lectura.

- **`FreeableMemory`**: este valor representa la cantidad de memoria sin utilizar que está disponible cuando la instancia de DocumentDB sin servidor se escala a su capacidad máxima. Por cada DCU en cuya capacidad actual esté por debajo de la capacidad máxima, este valor aumenta aproximadamente 2 GiB. Por lo tanto, esta métrica no se aproxima a cero hasta que la instancia se escala verticalmente al punto máximo posible.

  Si esta métrica se aproxima a un valor de cero, la instancia se ha escalado verticalmente tanto como es posible y se acerca al límite de memoria disponible. Considere aumentar la configuración máxima de DCU para el clúster. Si esta métrica se aproxima a un valor de cero en una instancia de lectura, considere añadir instancias de lectura adicionales al clúster. De esta forma, puede distribuir la parte de solo lectura de la carga de trabajo entre más instancias, y así reducir el uso de memoria en cada instancia de lectura.

- **`TempStorageIops`**: el número de IOPS realizadas en el almacenamiento local adjuntas a la instancia. Incluye las IOPS de lecturas y escrituras. Esta métrica representa un recuento y se mide

una vez por segundo. Esta es una nueva métrica para DocumentDB sin servidor. Para obtener más información, consulte Supervisión de Amazon DocumentDB con CloudWatch.

- **`TempStorageThroughput`**: la cantidad de datos transferidos desde el almacenamiento local asociado a la instancia y hacia este. Esta métrica representa un número de bytes y se mide una vez por segundo. Esta es una nueva métrica para DocumentDB sin servidor. Para obtener más información, consulte Supervisión de Amazon DocumentDB con CloudWatch.

Por lo general, la mayoría de los escalados verticales a instancias de DocumentDB sin servidor se deben al uso de la memoria y la actividad de CPU. Las métricas `TempStorageIops` y `TempStorageThroughput` pueden ayudar a diagnosticar los raros casos en los que la actividad de red para transferencias entre la instancia y los dispositivos de almacenamiento locales es responsable de aumentos de capacidad no esperados. Para supervisar otra actividad de red, puede utilizar estas métricas existentes:

- `NetworkReceiveThroughput`
- `NetworkThroughput`
- `NetworkTransmitThroughput`
- `StorageNetworkReceiveThroughput`
- `StorageNetworkThroughput`
- `StorageNetworkTransmitThroughput`

Cómo se aplican las métricas sin servidor de DocumentDB a su CloudWatch AWS factura

Los cargos por servicio sin servidor de DocumentDB que figuran en su AWS factura se calculan en función de la misma `ServerlessDatabaseCapacity` métrica que puede supervisar. El mecanismo de facturación puede diferir del CloudWatch promedio calculado para esta métrica en los casos en los que utilice la capacidad sin servidor de DocumentDB solo durante una hora. También puede variar si los problemas del sistema hacen que la CloudWatch métrica no esté disponible durante períodos breves. Por lo tanto, es posible que veas un valor ligeramente diferente de DCU-hours en tu factura que si calculas el número tú mismo a partir del valor `ServerlessDatabaseCapacity` promedio.

Ejemplos de comandos de la CloudWatch CLI de Amazon para métricas sin servidor de DocumentDB

Los siguientes AWS CLI ejemplos muestran cómo puede supervisar las CloudWatch métricas más importantes relacionadas con la tecnología sin servidor de DocumentDB. En cada caso, sustituya la cadena `Value=` para el parámetro `--dimensions` con el identificador de su propia instancia de DocumentDB sin servidor.

En el siguiente ejemplo de Linux se muestran los valores de capacidad mínima, máxima y promedio de una instancia, medidos cada 10 minutos durante 1 hora. El comando de fecha de Linux especifica las horas de inicio y finalización en relación con la fecha y la hora en curso. La función `sort_by` en el parámetro `--query` ordena los resultados cronológicamente basándose en el 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
```

El siguiente ejemplo de Linux muestra la supervisión de la capacidad de una instancia en un clúster. Mide el uso de capacidad mínimo, máximo y promedio de una instancia. Las mediciones se toman una vez por hora durante un período de tres horas. Estos ejemplos utilizan la métrica `DCUUtilization`, que representa un porcentaje del límite superior en las DCU, en lugar de `ServerlessDatabaseCapacity`, que representa un número fijo de DCU. De esta forma, no necesita conocer los números reales de los valores de DCU mínima y máxima en el rango de capacidad. Puede ver porcentajes que oscilan entre 0 y 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
```

En el siguiente ejemplo de Linux se hacen mediciones similares a las anteriores. En este caso, las medidas corresponden a la métrica `CPUUtilization`. Las mediciones se toman cada 10 minutos durante un período de 1 hora. Los números representan el porcentaje de CPU disponible utilizada, en función de los recursos de CPU disponibles para la configuración de capacidad máxima de la instancia.

```
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
```

En el siguiente ejemplo de Linux se hacen mediciones similares a las anteriores. En este caso, las medidas corresponden a la métrica `FreeableMemory`. Las mediciones se toman cada 10 minutos durante un período de 1 hora.

```
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
```

Supervisión del rendimiento de DocumentDB sin servidor con Información sobre rendimiento

Puede utilizar Información sobre rendimiento para supervisar el rendimiento de las instancias de DocumentDB sin servidor. Para conocer los procedimientos de información del rendimiento, consulte Supervisión con información sobre rendimiento.

Los siguientes son contadores de Información sobre rendimiento nuevos aplicables a instancias de DocumentDB sin servidor:

- **`os.general.serverlessDBCapacity`**: la capacidad actual de la instancia, en DCU. El valor corresponde a la `ServerlessDatabaseCapacity` CloudWatch métrica de la instancia.
- **`os.general.dcuUtilization`**— El porcentaje de la capacidad actual respecto a la capacidad máxima configurada. El valor corresponde a la `DCUUtilization` CloudWatch métrica de la instancia.
- **`os.general.maxConfiguredDcu`**: la capacidad máxima que configuró para esta instancia de DocumentDB sin servidor. Se mide en DCU.
- **`os.general.minConfiguredDcu`**: la capacidad mínima que configuró para esta instancia de DocumentDB sin servidor. Se mide en DCU.

Para ver la lista completa de contadores de información de rendimiento, consulte Métricas de contador para Información sobre rendimiento.

Cuando se muestran los valores de vCPU para una instancia de DocumentDB sin servidor en la Información sobre rendimiento, estos valores representan estimaciones basadas en el valor de DCU de la instancia. En el intervalo predeterminado de un minuto, los valores fraccionarios de vCPU se redondean al número entero más cercano. En intervalos más largos, el valor de vCPU que se muestra es el promedio de los valores de vCPU enteros de cada minuto.

Uso de clústeres elásticos de Amazon DocumentDB

Los clústeres elásticos de Amazon DocumentDB admiten cargas de trabajo con millones de petabytes de capacidad de almacenamiento reads/writes por segundo. Los clústeres elásticos también simplifican la forma en que los desarrolladores interactúan con Amazon DocumentDB al eliminar la necesidad de elegir, administrar o actualizar instancias.

> **Note**
>
> Los clústeres elásticos utilizan un protocolo de cableado compatible con MongoDB 5.0. No son compatibles con la versión 8.0 del motor Amazon DocumentDB y no están disponibles en las regiones de China ni en. AWS GovCloud

Los Amazon DocumentDB Elastic Clusters fueron creados para:

- Proporcionar una solución para los clientes que buscan una base de datos que ofrezca una escala prácticamente ilimitada con amplias capacidades de consulta y compatibilidad con la API de MongoDB.
- Ofrecer a los clientes límites de conexión más altos y reducir el tiempo de inactividad provocado por la aplicación de parches.
- Seguir invirtiendo en una arquitectura nativa en la nube, elástica y líder en su clase para las cargas de trabajo de JSON.

Temas

- Casos de uso de clústeres elásticos
- Ventajas de los clústeres elásticos
- Disponibilidad regional y de versión del clúster elástico
- Limitaciones
- Clústeres elásticos de Amazon DocumentDB: cómo funcionan
- Introducción a los clústeres elásticos de Amazon DocumentDB
- Prácticas recomendadas para los clústeres elásticos de Amazon DocumentDB
- Administración de los clústeres elásticos de Amazon DocumentDB
- Activación del cifrado de datos en reposo de un clúster elástico de Amazon DocumentDB

- Service-linked funciones en los clústeres elásticos

Casos de uso de clústeres elásticos

Las bases de datos documentales son útiles para cargas de trabajo que requieren un esquema flexible que permita un desarrollo rápido e iterativo. Para ver ejemplos de casos de uso de Amazon DocumentDB, consulte Casos de uso de bases de datos documentales.

A continuación, se incluyen algunos ejemplos de casos de uso para los que los clúster de elasticidad puedan ofrecer importantes ventajas:

Perfiles de usuario

Como las bases de datos documentales tienen un esquema flexible, pueden almacenar documentos que tengan atributos y valores de datos diferentes a escala. Los clúster elásticos son una solución práctica para los perfiles online en los que diferentes usuarios proporcionan diferentes tipos de información. Suponga que sus aplicaciones admiten cientos de millones de perfiles de usuario. Para admitir dichas aplicaciones puede usar clústeres elásticos, ya que se pueden ampliar y reducir para soportar millones de escrituras y lecturas en estos perfiles de usuario. También puede reducir verticalmente de cara a las horas de menor actividad para disminuir los costos.

Administración de contenido y registros históricos

Para administrar eficazmente el contenido, debe poder recopilar y agrupar contenido de una variedad de orígenes y enviárselo al cliente. Debido a su esquema flexible, las bases de datos documentales son perfectas para recopilar y almacenar cualquier tipo de datos. Puede utilizarlas para crear e incorporar nuevos tipos de contenido, incluido el contenido generado por el usuario, como imágenes, comentarios, y vídeos. Con el tiempo, es posible que su base de datos requiera más espacio de almacenamiento. Con los clústeres elásticos, puede distribuir sus datos entre más volúmenes de almacenamiento, lo que le permite almacenar petabytes de datos en un solo clúster.

Ventajas de los clústeres elásticos

AWS integración de servicios

Los clústeres elásticos de Amazon DocumentDB se integran con otros AWS servicios del mismo modo que lo hace Amazon DocumentDB:

- Migración: puede usar AWS Database Migration Service (DMS) para migrar de MongoDB y otras bases de datos relacionales a clústeres elásticos de Amazon DocumentDB.
- Supervisión: puede supervisar el estado y el rendimiento de su clúster elástico con Amazon. CloudWatch
- Seguridad: puedes configurar la autenticación y la autorización a través de AWS Identity and Access Management (IAM) para administrar los recursos de tu clúster elástico, como crear y actualizar un clúster elástico, y usar Amazon VPC para establecer conexiones seguras VPC-only . No se admite el inicio de sesión mediante autenticación de IAM para los clústeres elásticos.
- Administración de datos: puedes utilizarlos AWS Glue para importar y exportar datos a from/to otros AWS servicios, como Amazon S3, Amazon Redshift y Amazon Service. OpenSearch

Disponibilidad regional y de versión del clúster elástico

Disponibilidad regional para clústeres elásticos

La siguiente tabla muestra Regiones de AWS dónde están disponibles actualmente los clústeres elásticos de Amazon DocumentDB y el punto final de cada región.

Nombre de la región	Region	Zonas de disponibilidad
Este de EE. UU. (Norte de Virginia)	`us-east-1`	5
Este de EE. UU. (Ohio)	`us-east-2`	3
Oeste de EE. UU. (Oregón)	`us-west-2`	3
Asia-Pacífico (Hong Kong)	`ap-east-1`	3
Asia-Pacífico (Mumbai)	`ap-south-1`	3
Asia-Pacífico (Seúl)	`ap-northeast-2`	3
Asia-Pacífico (Singapur)	`ap-southeast-1`	3
Asia-Pacífico (Sídney)	`ap-southeast-2`	3
Asia-Pacífico (Tokio)	`ap-northeast-1`	3

Nombre de la región	Region	Zonas de disponibilidad
Canadá (centro)	`ca-central-1`	3
América del Sur (São Paulo)	`sa-east-1`	3
Europa (Fráncfort)	`eu-central-1`	3
Europa (Irlanda)	`eu-west-1`	3
Europa (Londres)	`eu-west-2`	3
Europa (Milán)	`eu-south-1`	3
Europa (París)	`eu-west-3`	3

Disponibilidad de versiones

Los clústeres elásticos admiten el protocolo Wire compatible con MongoDB 5.0. Para ver las diferencias entre los clústeres basados en instancias de Amazon DocumentDB 4.0 y los clústeres elásticos, consulte Diferencias funcionales entre Amazon DocumentDB 4.0 y los clústeres elásticos. Nota: Los clústeres elásticos no son compatibles con Amazon DocumentDB 8.0

Limitaciones

Administración de clústeres elásticos

Esta versión no admite las siguientes funciones y capacidades de administración de clústeres:

- Puerto personalizado (solo se admite el 27017)
- Capacidad de usar los usuarios y roles de IAM para la autenticación en la base de datos
- Capacidad para crear clústeres globales
- Eventos de Amazon DocumentDB existentes y suscripción a eventos
- Partición por rangos
- Partición de colección existente
- Multi-field clave compartida

- Cambio de la clave de partición
- Point-in-time restaurar
- Clonación
- Información de rendimiento

> **Note**
>
> Para obtener información sobre las cuotas de clústeres elásticos, consulteCuotas de Amazon DocumentDB.

Operaciones de consulta y escritura

Los siguientes comandos y capacidades de las operaciones de consulta y escritura no se admiten en esta versión:

- Comandos DDL durante las operaciones de escalado
- Profiler
- Grupos de parámetros
- AWS Config
- AWS Backup

Gestión de colecciones e índices

Esta versión no admite las siguientes funciones y capacidades de índices y colecciones:

- Índices únicos
- Índices parciales
- Índices de texto
- Índices vectoriales
- Compresión de documentos

Administración y diagnóstico

Esta versión no admite los siguientes comandos y capacidades de administración y diagnóstico:

- AWS Secrets Manager
- Role-based-access-control (RBAC) funciones personalizadas.
- Al conectarse, no se admite Write Concern con un valor de 0.
- Cambio de subredes pertenecientes a una VPC que actualmente no está asignada a un clúster elástico existente.

Opt-in features

Esta versión no admite las siguientes características opcionales de Amazon DocumentDB:

- Transacciones ACID
- DDL/DML auditoría
- Change streams
- Comandos de sesión

Clústeres elásticos de Amazon DocumentDB: cómo funcionan

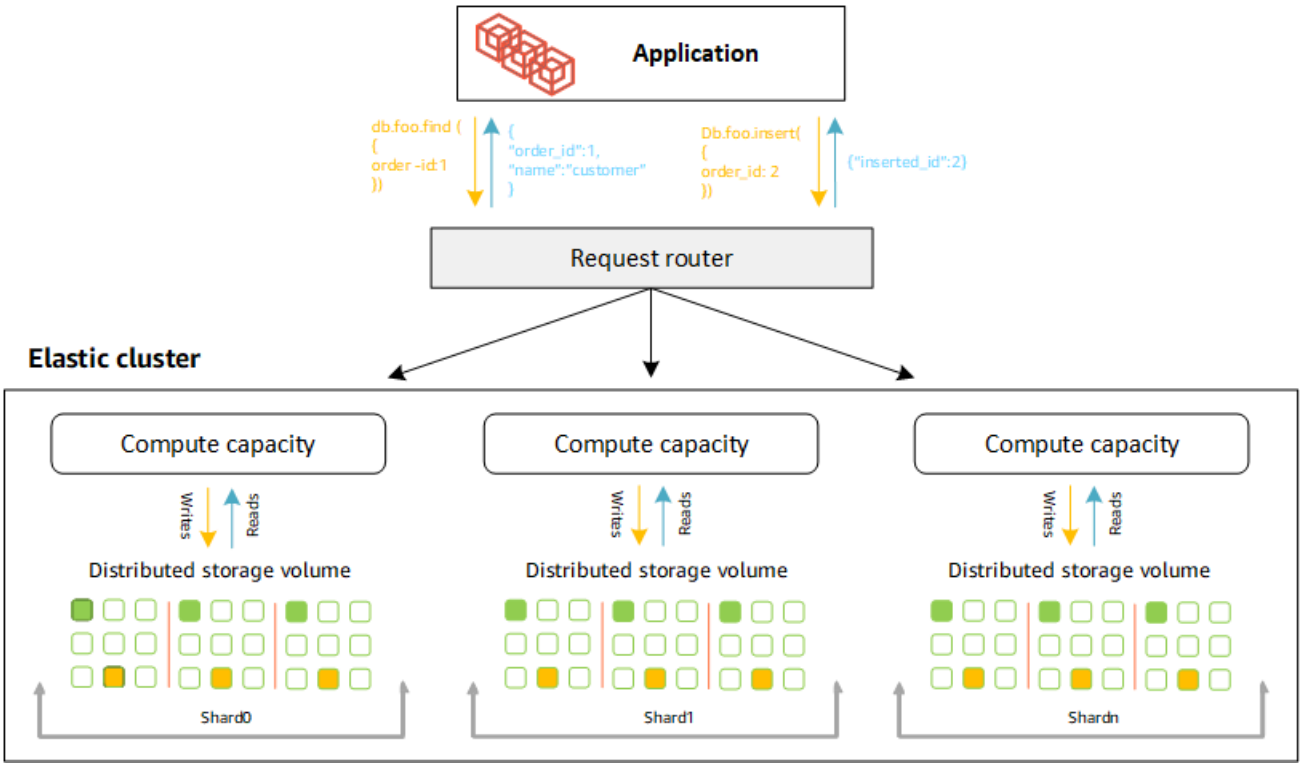
En los temas de esta sección se observa información acerca de los mecanismos y funciones que impulsan los clústeres elásticos de Amazon DocumentDB.

Temas

- Partición en los clústeres elásticos de Amazon DocumentDB
- Migración de clústeres elásticos
- Escalado de clústeres elásticos
- Fiabilidad de los clústeres elásticos
- Almacenamiento y disponibilidad de clústeres elásticos
- Diferencias funcionales entre Amazon DocumentDB 4.0 y los clústeres elásticos

Partición en los clústeres elásticos de Amazon DocumentDB

Los clústeres elásticos de Amazon DocumentDB utilizan particiones basadas en hash para hacer particiones de datos en un sistema de almacenamiento distribuido. La partición, también conocida como particionamiento, divide los conjuntos de datos grandes en conjuntos de datos pequeños en varios nodos, lo que le permite escalar horizontalmente la base de datos más allá de los límites de escalado vertical. Los clústeres elásticos utilizan la separación (o «disociación») del procesamiento y el almacenamiento en Amazon DocumentDB, lo que le permite a usted escalar de forma independiente. En lugar de volver a particionar las colecciones moviendo pequeñas particiones de datos entre los nodos de cómputo, los clústeres elásticos copian los datos de manera eficiente dentro del sistema de almacenamiento distribuido.



Definiciones de particiones

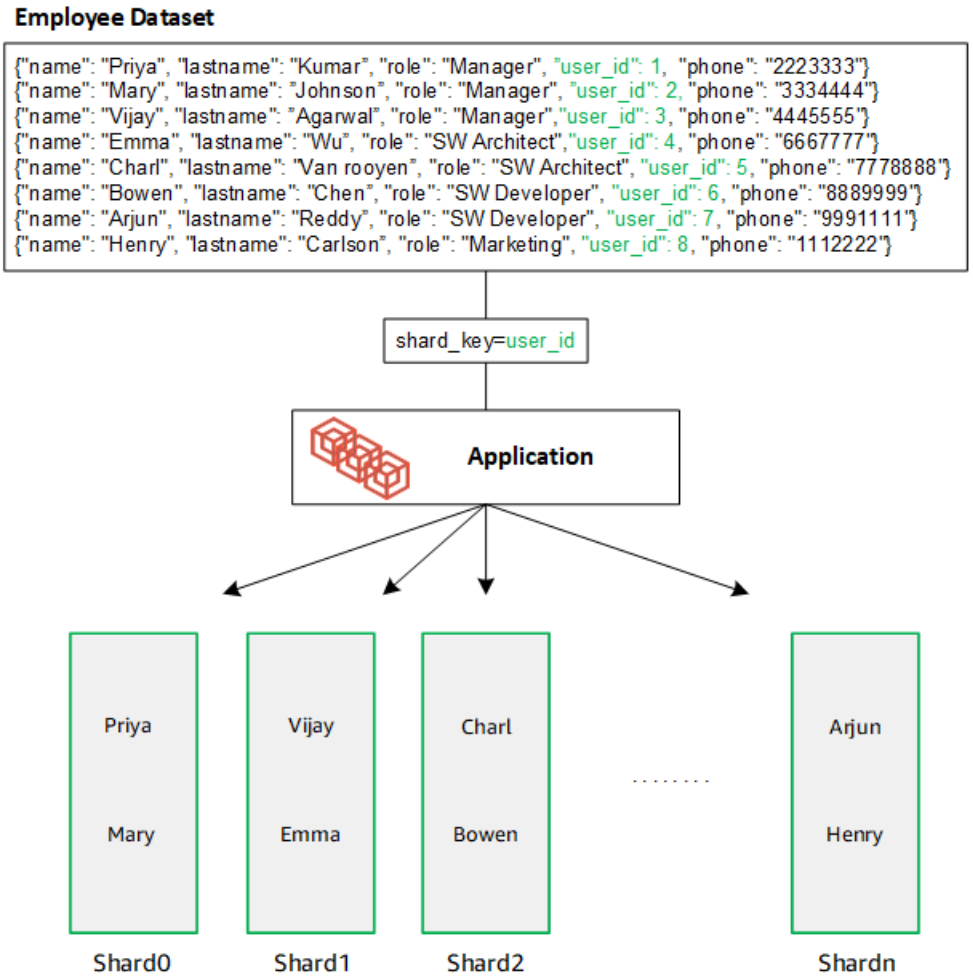
Definiciones de la nomenclatura de particiones:

- Partición: una partición proporciona el cómputo para un clúster elástico. Tendrá una sola instancia de escritura y de 0 a 15 réplicas de lectura. De forma predeterminada, una partición tendrá dos instancias: una de escritura y una réplica de lectura única. Puede configurar un máximo de 32 particiones y cada instancia de partición puede tener un máximo de 64 vCPU.

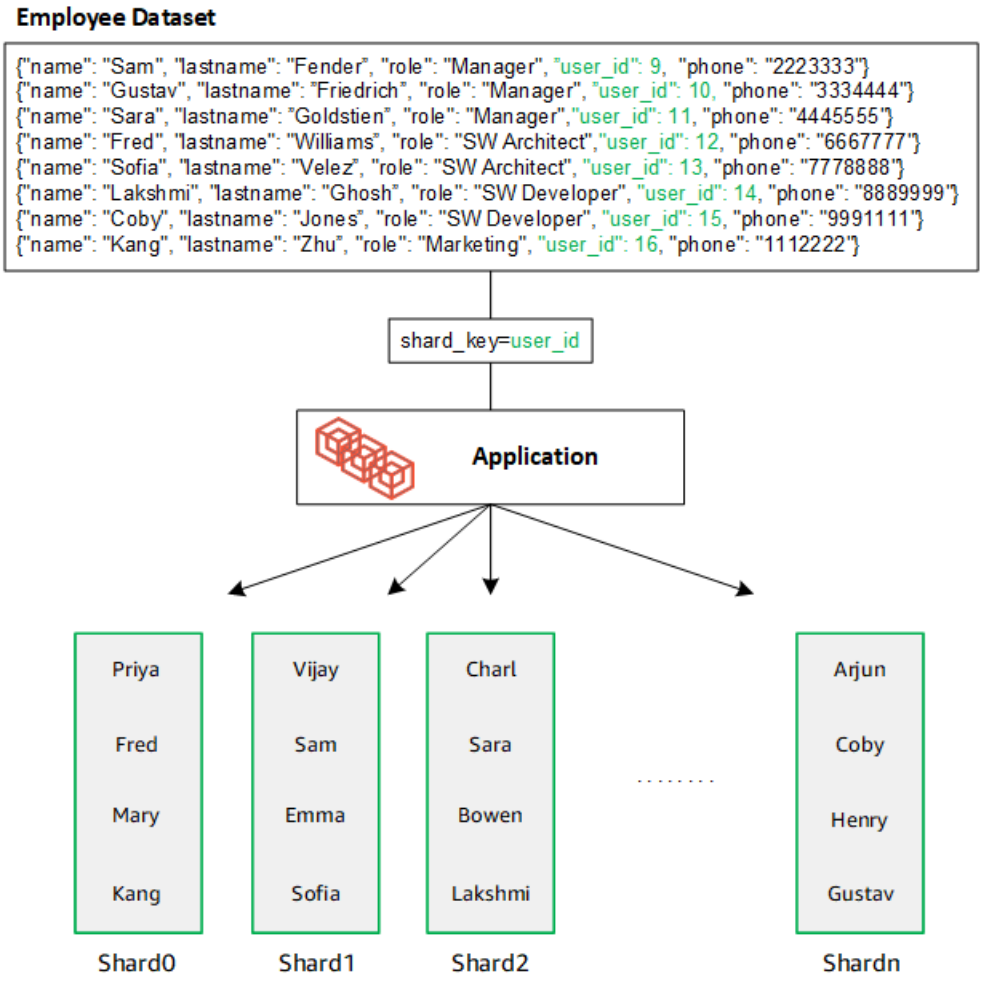
- Clave de partición: una clave de partición es un campo obligatorio en los documentos JSON de las colecciones con particiones, que los clústeres elásticos utilizan para distribuir el tráfico de lectura y escritura a la partición correspondiente.
- Colección con particiones: en una colección con particiones, los datos se distribuyen dentro de un clúster elástico en particiones de datos.
- Partición: una partición es una parte lógica de los datos fragmentados. Al crear una colección con particiones, los datos se organizan automáticamente en particiones dentro de cada partición en función de la clave de la partición. Cada fragmento tiene varias particiones.

Distribución de los datos entre las particiones configuradas

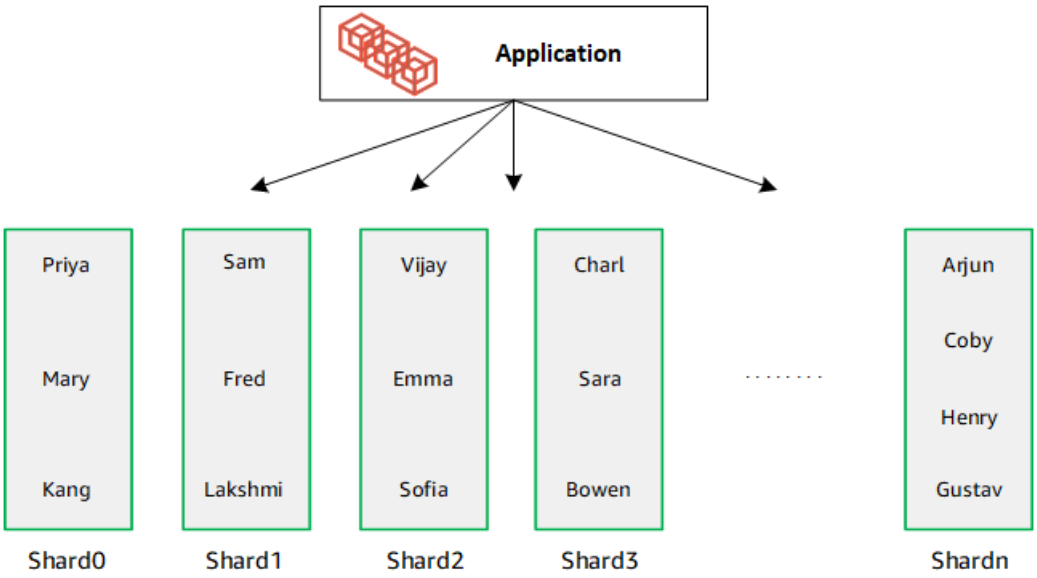
Cree una clave de partición que tenga muchos valores únicos. Una buena clave de partición particionará uniformemente sus datos entre las particiones subyacentes, proporcionando a su carga de trabajo el mejor rendimiento y desempeño. En el siguiente ejemplo, se muestran los datos del nombre de un empleado que utilizan una clave de partición denominada “user_id”:



DocumentDB utiliza la partición hash para particionar los datos en las particiones subyacentes. Los datos adicionales se insertan y distribuyen de la misma manera:



Cuando escala horizontalmente la base de datos añadiendo particiones adicionales, Amazon DocumentDB redistribuye automáticamente los datos:



Migración de clústeres elásticos

Amazon DocumentDB admite la migración de datos con particiones de MongoDB a clústeres elásticos. Se admiten métodos de migración sin conexión, en línea e híbridos. Para obtener más información, consulte Migración y actualización de Amazon DocumentDB.

Escalado de clústeres elásticos

Los clústeres elásticos de Amazon DocumentDB ofrecen la posibilidad de aumentar el número de particiones (escalado horizontal) en el clúster elástico y el número de vCPU aplicadas a cada partición (escalado vertical). También puede reducir la cantidad de particiones y la capacidad de cómputo (vCPU) según sea necesario.

Para conocer las prácticas recomendadas de escalado, consulte Escalado de clústeres elásticos.

> **Note**
>
> Cluster-level el escalado también está disponible. Para obtener más información, consulte Escalado de clústeres de Amazon DocumentDB.

Fiabilidad de los clústeres elásticos

Amazon DocumentDB está diseñado para ofrecer fiabilidad, durabilidad y tolerancia a errores. Para mejorar la disponibilidad, los clústeres elásticos implementan dos nodos por partición ubicados en

diferentes zonas de disponibilidad. Amazon DocumentDB también incluye varias características automáticas que la convierten en una solución de base de datos de confianza. Para obtener más información, consulte Fiabilidad de Amazon DocumentDB.

Almacenamiento y disponibilidad de clústeres elásticos

Los datos de Amazon DocumentDB se almacenan en el volumen del clúster, que es un volumen único y virtual que usa unidades de estado sólido (SSD). Un volumen de clúster consta de seis copias de los datos, que se replican automáticamente en varias zonas de disponibilidad de una sola AWS región. Esta replicación ayuda a garantizar que los datos se conserven durante mucho tiempo, con menos riesgo de que se pierdan los datos. También ayuda a garantizar que el clúster esté más disponible durante una conmutación por error, porque ya existen copias de sus datos en otras zonas de disponibilidad. Para obtener más información sobre el almacenamiento, la alta disponibilidad y la replicación, consulte Funcionamiento de Amazon DocumentDB.

Diferencias funcionales entre Amazon DocumentDB 4.0 y los clústeres elásticos

Existen las siguientes diferencias funcionales entre Amazon DocumentDB 4.0 y los clústeres elásticos.

- El resultado de `top` y `collStats` se fragmenta en particiones. En el caso de las colecciones con particiones, los datos se distribuyen entre varias particiones y a los informes de `collStats` se agrega `collScans` desde las particiones.
- Las estadísticas de recopilación de `top` y `collStats` para las colecciones con particiones se restablecen cuando se cambia el recuento de particiones del clúster.
- La función de copia de seguridad integrada ahora es compatible con `serverStatus`. Acción: los desarrolladores y las aplicaciones con función de backup pueden recopilar estadísticas sobre el estado del clúster de Amazon DocumentDB.
- El campo `SecondaryDelaySecs` reemplaza a `slaveDelay` en la salida `replSetGetConfig`.
- El comando `hello` reemplaza a `isMaster`. `hello` devuelve un documento que describe la función del clúster elástico.
- El `$elemMatch` operador de los clústeres elásticos solo hace coincidir los documentos del primer nivel de anidación de una matriz. En Amazon DocumentDB 4.0, el operador recorre todos los niveles antes de devolver los documentos coincidentes. Por ejemplo:

```
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 proyección “$” de Amazon DocumentDB 4.0 devuelve todos los documentos con todos los campos. Con los clústeres elásticos, el comando `find` con una proyección “$” devuelve los documentos que coinciden con el parámetro de consulta y contienen solo el campo que coincide con la proyección “$”.
- En los clústeres elásticos, los comandos `find` con parámetros de consulta `$regex` y `$options` devuelven un error: «No se pueden establecer opciones tanto en $regex como en $options».

- Con los clústeres elásticos, `$indexOfCP` ahora devuelve “-1” cuando:
  - la subcadena no se encuentra en `string expression`, o
  - `start` es un número mayor que `end`, o
  - `start` es un número mayor que la longitud en bytes de la cadena.

  En Amazon DocumentDB 4.0, `$indexOfCP` devuelve “0” cuando la posición `start` es un número mayor que `end` o que la longitud en bytes de la cadena.

- Con los clústeres elásticos, las operaciones de proyección en `_id fields`, por ejemplo, `{"_id.nestedField" : 1}`, devuelven documentos que solo incluyen el campo proyectado. Mientras tanto, en Amazon DocumentDB 4.0, los comandos de proyección de campos anidados no filtran ningún documento.

Introducción a los clústeres elásticos de Amazon DocumentDB

En esta sección de introducción se explica cómo crear y consultar su primer clúster elástico.

Existen muchas formas de conectarse a Amazon DocumentDB y empezar a usarlo. El siguiente procedimiento es la forma más rápida, sencilla y fácil de que los usuarios comiencen a utilizar nuestra potente base de datos de documentos. En esta guía se utiliza el AWS CloudShell para conectarse al clúster de Amazon DocumentDB directamente desde la Consola de administración de AWS y consultarlo. Los nuevos clientes que cumplan los requisitos para la capa AWS gratuita pueden usar Amazon DocumentDB de forma CloudShell gratuita. Si su AWS CloudShell entorno o clúster elástico de Amazon DocumentDB utiliza más recursos que los de la capa gratuita, se le cobrarán AWS las tarifas normales por esos recursos. Esta guía le ayudará a empezar a utilizar Amazon DocumentDB en menos de 5 minutos.

Temas

- Requisitos previos
- Paso 1: crear un clúster elástico
- Paso 2: conectarse a su clúster elástico
- Paso 3: hacer una partición de su colección; insertar y consultar los datos
- Paso 4: Explorar

Requisitos previos

Antes de crear el primer clúster de Amazon DocumentDB, debe hacer lo siguiente:

Creación de una cuenta de Amazon Web Services (AWS)

Para empezar a utilizar Amazon DocumentDB, debe tener una cuenta de Amazon Web Services (AWS). La AWS cuenta es gratuita. Solo se paga por los servicios y los recursos que se utilicen.

Si no tiene una Cuenta de AWS, complete los siguientes pasos para crear una.

Para inscribirse en un Cuenta de AWS

1. https://signin.aws.amazon.com/signup¿Abierto? request_type=registrarse.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica o mensaje de texto e indicar un código de verificación en el teclado del teléfono.

   Cuando te registras en un, se crea un Cuenta de AWS. Usuario raíz de la cuenta de AWS El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar Tareas que requieren acceso de usuario raíz.

Configura los permisos necesarios AWS Identity and Access Management (IAM).

El acceso para administrar los recursos de Amazon DocumentDB, como los clústeres, las instancias y los grupos de parámetros de clústeres, requiere credenciales que AWS se puedan utilizar para autenticar las solicitudes. Para obtener más información, consulte Identity and Access Management para Amazon DocumentDB.

1. En la barra de búsqueda de Consola de administración de AWS, introduzca **`IAM`** y seleccione IAM en el menú desplegable.
2. Cuando esté en la consola de IAM, seleccione Usuarios en el panel de navegación.
3. Seleccione su nombre de usuario.
4. Elija Añadir permisos.
5. Seleccione Adjuntar políticas directamente.
6. Escriba `AmazonDocDBElasticFullAccess` en la barra de búsqueda y selecciónelo en cuanto aparezca en los resultados de búsqueda.
7. Elija Siguiente.
8. Elija Añadir permisos.

> **Note**
>
> Tu AWS cuenta incluye una VPC predeterminada en cada región. Si decide utilizar una Amazon VPC, complete los pasos del tema Creación de una Amazon VPC en la Guía del usuario de Amazon VPC.

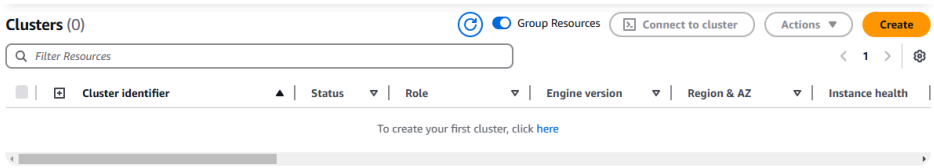
Paso 1: crear un clúster elástico

En esta sección, explicamos cómo crear un clúster elástico completamente nuevo, mediante las instrucciones siguientes Consola de administración de AWS o AWS CLI siguiendo estas instrucciones.

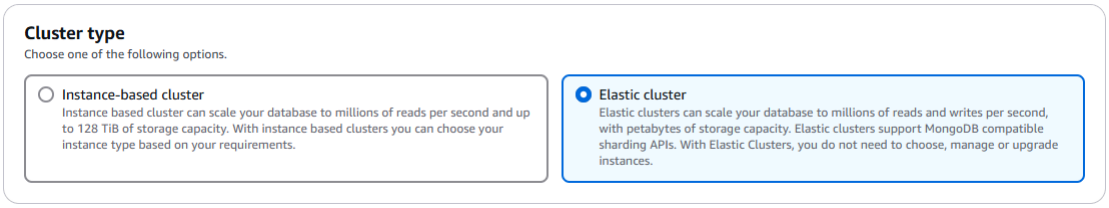
Using the Consola de administración de AWS

Para crear una configuración de clúster elástico mediante la Consola de administración de AWS:

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En la consola de administración de Amazon DocumentDB, en Clústeres, elija Crear.



3. En la página Crear clúster de Amazon DocumentDB, en la sección Tipo de clúster, elija Clúster elástico.



4. En la sección Configuración, configure lo siguiente:

   a. En el campo Nombre del clúster, introduzca un identificador de clúster único (siguiendo los requisitos de denominación que aparecen debajo del campo).

   b. En el campo Recuento de particiones, introduzca el número de particiones que desea tener en el clúster. El número máximo de particiones por clúster es 32.

> **Note**
>
> Se implementarán dos nodos para cada partición. Ambos nodos tendrán la misma capacidad de partición.

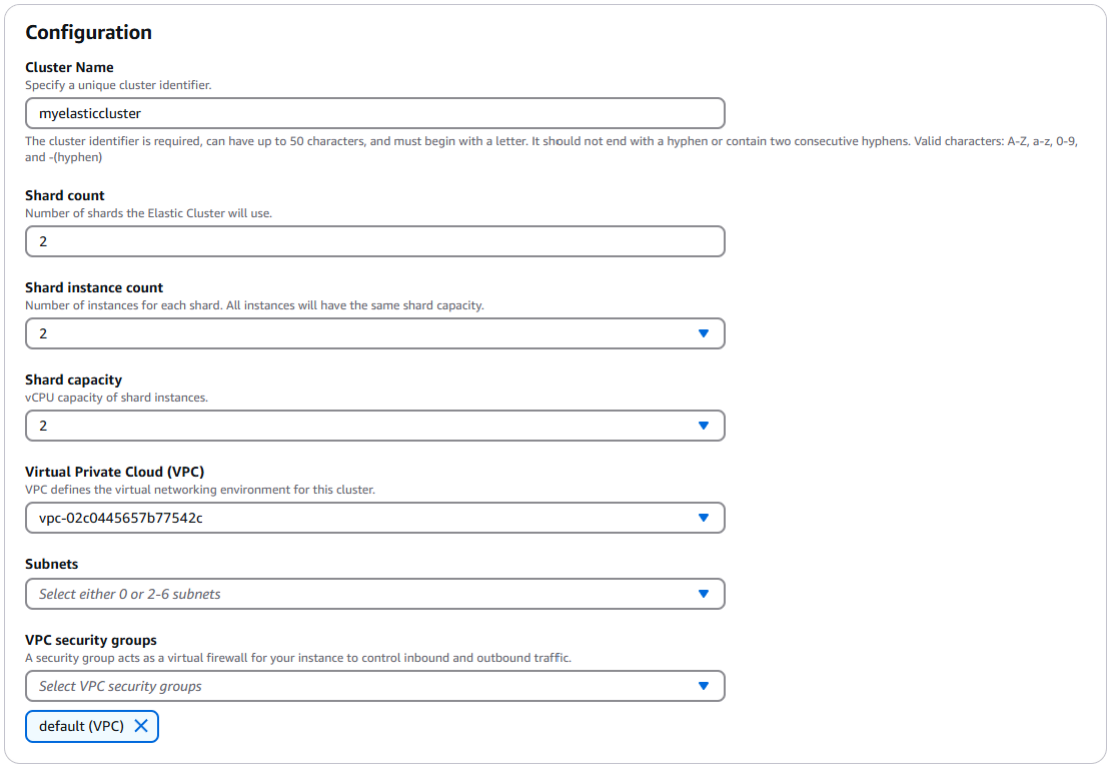
c. En el campo Recuento de instancias de partición, elija el número de instancias de réplica que desee asociar a cada partición. El número máximo de instancias de partición es 16, en incrementos de 1. Todas las instancias de réplica tienen la misma capacidad de partición que se define en el campo siguiente. Para fines de prueba, el valor predeterminado de 2 debería ser suficiente.

> **Note**
>
> El número de instancias de réplica se aplica a todas las particiones del clúster elástico. Un valor de recuento de instancias de particiones igual a 1 significa que hay una instancia de escritura y las instancias adicionales son réplicas que se pueden usar para llevar a cabo lecturas y mejorar la disponibilidad. Para fines de prueba, el valor predeterminado de 2 debería ser suficiente.

d. En el campo Capacidad de la partición, elija el número de CPU virtuales (vCPU) que desee asociar a cada instancia de partición. La cantidad máxima de vCPU por instancia de partición es 64. Los valores permitidos son 2, 4, 8, 16, 32, 64. Para fines de prueba, el valor predeterminado de 2 debería ser suficiente.

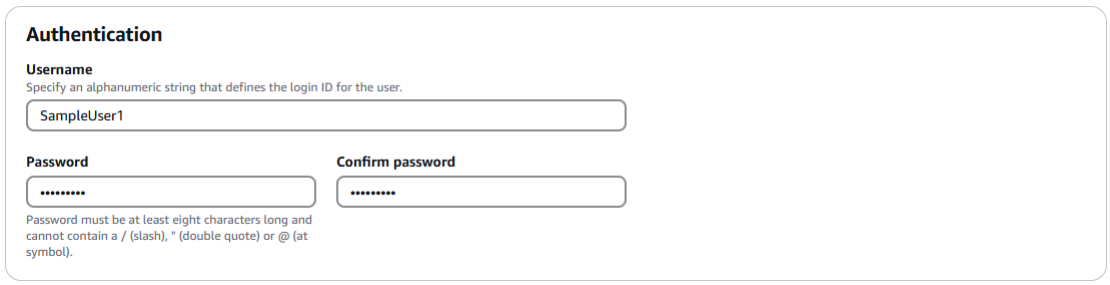
e. En el campo Nube privada virtual (VPC), elija una VPC de la lista desplegable.

f. Para las subredes y los grupos de seguridad de VPC, puede usar los valores predeterminados o seleccionar tres subredes de su elección y hasta tres grupos de seguridad de VPC (uno como mínimo).



5. En la sección Autenticación, ingrese una cadena que identifique el nombre de inicio de sesión del usuario principal en el campo Nombre de usuario.

   En el campo Contraseña, introduzca una contraseña única que cumpla con las instrucciones y, a continuación, confírmela.



6. En la sección Cifrado, mantenga la configuración predeterminada (Clave predeterminada).

   Si lo desea, puede introducir un AWS KMS key ARN que haya creado. Para obtener más información, consulte Activación del cifrado de datos en reposo de un clúster elástico de Amazon DocumentDB.

> **Important**
>
> El cifrado debe estar habilitado para los clústeres elásticos.

7. En la sección Copia de seguridad, edite los campos de acuerdo con los requisitos de copia de seguridad. Para fines de prueba, puede retener la configuración predeterminada.



   a. Periodo de retención de copias de seguridad: en la lista, elija el número de días que se deben conservar las copias de seguridad automáticas de este clúster antes de eliminarlas.

   b. Periodo de copia de seguridad: especifique la hora del día en que Amazon DocumentDB debe hacer las copias de seguridad de este clúster y la duración de las mismas.

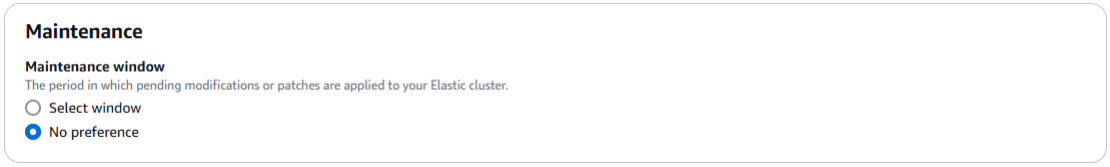
      i. Elija Seleccionar periodo si desea configurar la hora y la duración de la creación de las copias de seguridad.

         Hora de inicio: en la primera lista, elija la hora de inicio (UTC) para las copias de seguridad automáticas. En la segunda lista, elija el minuto de la hora en que desea que comiencen las copias de seguridad automáticas.

         Duración: en la lista, elija el número de horas que se deben asignar para crear las copias de seguridad automáticas.

      ii. Seleccione Sin preferencia si desea que Amazon DocumentDB elija la hora y la duración de la creación de las copias de seguridad.

8. En la sección Mantenimiento, elija el día, la hora y la duración en que se aplicarán las modificaciones o los parches al clúster. Para fines de prueba, puede retener la configuración predeterminada.



9. Elija Create cluster.

El clúster elástico se está aprovisionando ahora. Esto puede tardar unos minutos en terminar. Puede conectarse a su clúster cuando el estado del clúster elástico aparezca como Disponible en la lista de Clústeres.

Using the AWS CLI

Para crear un clúster elástico mediante AWS CLI, utilice la `create-cluster` operación con los siguientes parámetros:

- --cluster-name: obligatorio. El nombre actual del clúster de escala elástica tal como se ingresó durante la creación o se modificó por última vez.
- --shard-capacity: obligatorio. El número de vCPU asignado a cada partición. El máximo es 64. Los valores permitidos son 2, 4, 8, 16, 32, 64.
- --shard-count: obligatorio. La cantidad de particiones asignadas al clúster. El máximo es 32.
- --shard-instance-count: opcional. El número de instancias de réplica que se aplican a todos los fragmentos de este clúster. El máximo es 16.
- --admin-user-name: obligatorio. Nombre del usuario asociado al usuario administrador.
- --admin-user-password: obligatorio. La contraseña asociada al usuario administrador.
- --auth-type: obligatorio. El tipo de autenticación utilizado para determinar dónde buscar la contraseña que se usa para acceder al clúster elástico. Los tipos válidos son `PLAIN_TEXT` o `SECRET_ARN`.
- --vpc-security-group-ids: opcional. Configure una lista de grupos de seguridad de la VPC de EC2 para asociar a este clúster.
- --preferred-maintenance-window: opcional. Configure el tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en tiempo universal coordinado (UTC).

  El formato es: `ddd:hh24:mi-ddd:hh24:mi`. Días válidos: lunes, martes, miércoles, jueves, viernes, sábado, domingo

  El valor predeterminado es un periodo de 30 minutos seleccionado al azar de un bloque de 8 horas de tiempo para cada región de Amazon Web Services, que tiene lugar un día de la semana de forma aleatoria.

  El plazo mínimo es 30 minutos.
- --kms-key-id: opcional. Configure el identificador de la clave de KMS de un clúster cifrado.

El identificador de la clave de KMS es el nombre de recurso de Amazon (ARN) de la clave de AWS KMS cifrado. Si está creando un clúster con la misma cuenta de Amazon Web Services a la que pertenece la clave de cifrado de KMS utilizada para cifrar el clúster nuevo, puede utilizar el alias de la clave de KMS en lugar del ARN para la clave de cifrado de KMS.

Si no se especifica ninguna clave de cifrado KmsKeyId y el `StorageEncrypted` parámetro es verdadero, Amazon DocumentDB utiliza la clave de cifrado predeterminada.

- --preferred-backup-window: opcional. El intervalo de tiempo diario preferido durante el que se crean las copias de seguridad automatizadas. La ventana predeterminada es una ventana de 30 minutos seleccionada al azar de un bloque de 8 horas para cada una. Región de AWS
- --backup-retention-period: opcional. El número de días durante los que se retienen las copias de seguridad automatizadas. El valor predeterminado es 1.
- --storage-encrypted: opcional. Configura si el clúster está cifrado o no.

  `--no-storage-encrypted` especifica que el clúster no está cifrado.
- --subnet-ids: opcional. Configure los identificadores de subred de la red.

Para utilizar el ejemplo siguiente, sustituya *`user input placeholder`* con su propia información.

> **Note**
>
> Los siguientes ejemplos incluyen la creación de una clave KMS específica. Para usar la clave KMS predeterminada, no incluya el `--kms-key-id` parámetro.

Para 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
```

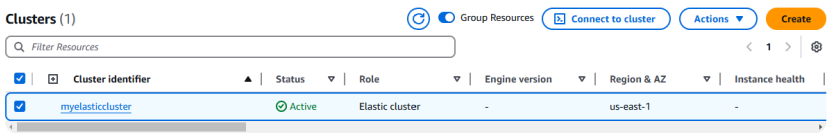
Para Windows:

```
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
```

Paso 2: conectarse a su clúster elástico

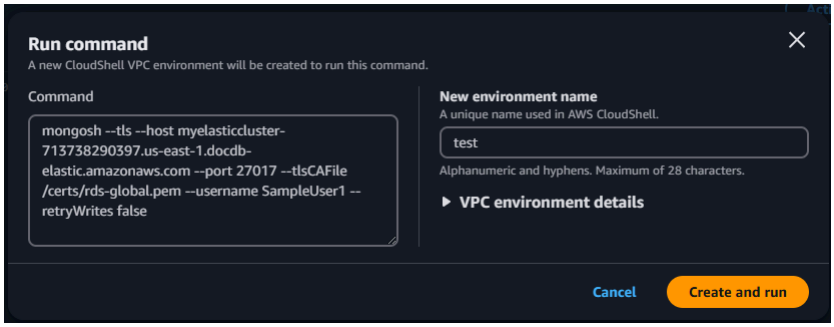
Conéctese a su clúster elástico de Amazon DocumentDB mediante AWS CloudShell.

1. En la consola de administración de Amazon DocumentDB, en Clústeres, localice el clúster elástico que ha creado. Para elegir el clúster, haga clic en la casilla de verificación situada junto a él.



2. Seleccione Conectarse al clúster (que se encuentra junto al menú desplegable Acciones). Este botón se activa solamente después de haber hecho clic en la casilla de verificación situada junto al clúster y el estado del clúster se muestra como Disponible. Aparece la pantalla de comandos CloudShell Ejecutar.

3. En el campo Nombre del nuevo entorno, ingrese un nombre único, como “prueba” y elija Crear y ejecutar. Los detalles del entorno de la VPC se configuran automáticamente para la base de datos de Amazon DocumentDB.



4. Cuando vea la petición, introduzca la contraseña que creó en el Paso 1: crear un clúster elástico de Amazon DocumentDB (subpaso 5).



Después de ingresar la contraseña y de que la petición se convierte en `direct: mongos] <env-name>>`, significa que se ha conectado correctamente al clúster de Amazon DocumentDB.

> **Note**
>
> Para obtener información sobre la solución de problemas, consulte Solución de problemas de Amazon DocumentDB.

Paso 3: hacer una partición de su colección; insertar y consultar los datos

Los clústeres elásticos aportan compatibilidad para crear particiones en Amazon DocumentDB. Ahora que está conectado a su clúster, puede hacer particiones, insertar datos y ejecutar algunas consultas.

1. Para hacer particiones en una colección, escriba lo siguiente:

```
sh.shardCollection("db.Employee1" , { "Employeeid" : "hashed" })
```

2. Para insertar un solo documento, escriba lo siguiente:

```
db.Employee1.insertOne({"Employeeid":1, "Name":"Joe", "LastName": "Bruin", "level":
 1 })
```

Se muestra lo siguiente:

```
WriteResult({ "nInserted" : 1 })
```

3. Para leer el documento que escribió, introduzca el comando `findOne()` (devuelve un único documento):

```
db.Employee1.findOne()
```

Se muestra lo siguiente:

```
{
"_id" : ObjectId("61f344e0594fe1a1685a8151"),
"EmployeeID" : 1,
"Name" : "Joe",
"LastName" : "Bruin",
"level" : 1
}
```

4. Para realizar algunas consultas más, plantéese un caso de uso de perfil de juegos. Primero, inserte algunas entradas en una colección titulada “Empleado”. Introduzca lo siguiente:

```
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 }
])
```

Se muestra lo siguiente:

```
{ acknowledged: true,
```

```
     insertedIds: {
        '0': ObjectId('679d02cd6b5a0581be78bcbd'),
        '1': ObjectId('679d02cd6b5a0581be78bcbe'),
        '2': ObjectId('679d02cd6b5a0581be78bcbf'),
        '3': ObjectId('679d02cd6b5a0581be78bcc0')
    }
}
```

5. Para devolver todos los documentos de la colección de perfiles, introduzca el comando `find` ():

```
db.Employee.find()
```

   Se muestran los datos que ingresó en el paso 4.

6. Para consultar un solo documento, incluya un filtro (por ejemplo: “Katie”). Introduzca lo siguiente:

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

   Se muestra lo siguiente:

```
[

   {

     _id: ObjectId('679d02cd6b5a0581be78bcc0'),

     Employeeid: 4,

     name: 'Katie',

     lastname: 'Schaper',

     level: 3

   }

]
```

7. Para buscar un perfil y modificarlo, introduzca el comando `findAndModify`. En este ejemplo, al empleado “Matt” se le asigna un nivel superior de “14”:

```
db.Employee.findAndModify({
```

```
    query: { "Employeeid" : 1, "name" : "Matt"},
    update: { "Employeeid" : 1, "name" : "Matt", "lastname" : "Winkle", "level" :
 14 }
})
```

Aparece el siguiente resultado (tenga en cuenta que el nivel no ha cambiado todavía):

```
{

   _id: ObjectId('679d02cd6b5a0581be78bcbd'),

   Employeeid: 1,

   name: 'Matt',

   lastname: 'Winkle',

   level: 12

}
```

8. Para verificar el aumento de nivel, introduzca la siguiente consulte:

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

Se muestra lo siguiente:

```
[
   {

   _id: ObjectId('679d02cd6b5a0581be78bcbd'),

   Employeeid: 1,

   name: 'Matt',

   lastname: 'Winkle',

   level: 14

   }
]
```

Paso 4: Explorar

¡Enhorabuena! Ha completado correctamente el procedimiento de introducción a los clústeres elásticos de Amazon DocumentDB.

Pasos siguientes Descubra cómo aprovechar al máximo esta base de datos con algunas de sus características más populares:

- Prácticas recomendadas para los clústeres elásticos de Amazon DocumentDB
- Administración de los clústeres elásticos de Amazon DocumentDB

> **Note**
>
> El clúster elástico que creó a partir de este procedimiento de introducción seguirá acumulando costos a menos que lo elimine. Para obtener instrucciones, consulte Eliminación de un clúster elástico.

Prácticas recomendadas para los clústeres elásticos de Amazon DocumentDB

Descubra las mejores prácticas para trabajar con clústeres elásticos de Amazon DocumentDB. Todas las mejores recomendadas para los clústeres de Amazon DocumentDB basados en instancias también se aplican a los clústeres elásticos. Esta sección se actualiza continuamente a medida que se identifican nuevas prácticas recomendadas.

Temas

- Elección de las claves de partición
- Administración de conexiones
- Colecciones sin particiones
- Escalado de clústeres elásticos
- Monitoreo de clústeres elásticos

Elección de las claves de partición

En la siguiente lista se describen las pautas para crear claves de partición.

- Utilice una clave hash distribuida uniformemente para distribuir los datos en todos las particiones del clúster (evite las teclas de acceso rápido).
- Usa tu clave de partición en todas las solicitudes read/update /delete para evitar las consultas de recopilación dispersa.
- Evite las claves fragmentadas anidadas al realizar las operaciones /delete. read/update
- Al realizar operaciones por lotes, establezca `ordered` en falso para que todos las particiones puedan ejecutarse en paralelo y mejorar las latencias.

Administración de conexiones

En la siguiente lista se describen las pautas para administrar las conexiones con la base de datos.

- Supervise el número de conexiones y la frecuencia con la que se abren y cierran nuevas conexiones.
- Distribuya sus conexiones en todas las subredes de la configuración de su aplicación. Si su clúster está configurado en varias subredes, pero solo utiliza un subconjunto de las subredes, es posible que el número máximo de conexiones sea limitado.

Colecciones sin particiones

A continuación, se describe una guía para las colecciones sin particiones.

- Cuando trabaje con colecciones sin particiones, al distribuir la carga intente mantener las colecciones sin particiones que se usen mucho, en diferentes bases de datos. Los clústeres elásticos de Amazon DocumentDB colocan las bases de datos en diferentes particiones y comparten la ubicación de colecciones sin particiones de la misma base de datos en la misma partición.

Escalado de clústeres elásticos

En la siguiente lista se describen las pautas para escalar los clústeres elásticos.

- Las operaciones de escalado pueden provocar un breve período de errores intermitentes en la base de datos y en la red. Siempre que sea posible, evite escalar durante las horas pico. Intente escalar durante los periodos de mantenimiento.

- Es preferible aumentar y reducir la capacidad de las particiones (cambiar el recuento de vCPU por partición) para aumentar el cómputo en lugar de aumentar o disminuir el recuento de particiones, ya que es más rápido y tiene una duración más corta de los errores intermitentes de la base de datos y de la red.
- Al anticipar el crecimiento, opte por aumentar el número de particiones en lugar de escalar la capacidad de las particiones. Esto le permite escalar su clúster aumentando la capacidad de las particiones en situaciones en las que necesite escalar rápidamente.
- Supervise las políticas de reintentos del cliente y vuelva a intentarlo con retrocesos exponenciales y fluctuaciones para evitar sobrecargar la base de datos cuando se produzcan errores al escalar.
- Cambiar el número de fragmentos puede llevar tiempo, ya que los datos se deben mover de forma segura de un fragmento a otro. Para minimizar el riesgo de errores, no detenga ni realice ningún otro cambio en el estado del clúster mientras el cambio esté en curso.

Monitoreo de clústeres elásticos

En la siguiente lista se describen las pautas para monitorear los clústeres elásticos.

- Realiza un seguimiento de la relación entre el máximo y el promedio de tus métricas por fragmento para determinar si estás generando un tráfico irregular (si tienes un punto crítico). key/hot Las métricas clave para hacer un seguimiento de las proporciones entre picos y promedios son las siguientes:
  - `PrimaryInstanceCPUUtilization`
    - Esto se puede monitorear a nivel de cada partición.
    - A nivel de clúster, puede monitorear la desviación media de p99.
  - `PrimaryInstanceFreeableMemory`
    - Esto se puede monitorear a nivel de cada partición.
    - A nivel de clúster, puede monitorear la desviación media de p99.
  - `DatabaseCursorsMax`
    - Esto se debe supervisar a nivel de partición para determinar el sesgo.
  - `Documents-Inserted/Updated/Returned/Deleted`
    - Esto se debe supervisar a nivel de partición para determinar el sesgo.

Administración de los clústeres elásticos de Amazon DocumentDB

Para administrar un clúster de Amazon DocumentDB, debe disponer de una política de IAM con los permisos del plano de control de Amazon DocumentDB. Estos permisos le permiten crear, modificar y eliminar clústeres. La DocumentDBFullAccess política de Amazon proporciona todos los permisos necesarios para administrar un clúster elástico de Amazon DocumentDB.

En los temas siguientes se muestra cómo realizar diversas tareas al trabajar con clústeres elásticos de Amazon DocumentDB.

Temas

- Modificación de configuraciones de clústeres elásticos
- Monitoreo de un clúster elástico
- Eliminación de un clúster elástico
- Administración de instantáneas de los clústeres elásticos
- Detención e inicio de un clúster elástico de Amazon DocumentDB
- Mantenimiento de clústeres elásticos de Amazon DocumentDB

Modificación de configuraciones de clústeres elásticos

En esta sección, explicamos cómo modificar un clúster elástico mediante las instrucciones siguientes Consola de administración de AWS o siguiendo AWS CLI las instrucciones siguientes.

Uno de los usos principales de la modificación del clúster es escalar los fragmentos aumentando o disminuyendo la capacidad de cálculo de los fragmentos por recuento de and/or fragmentos.

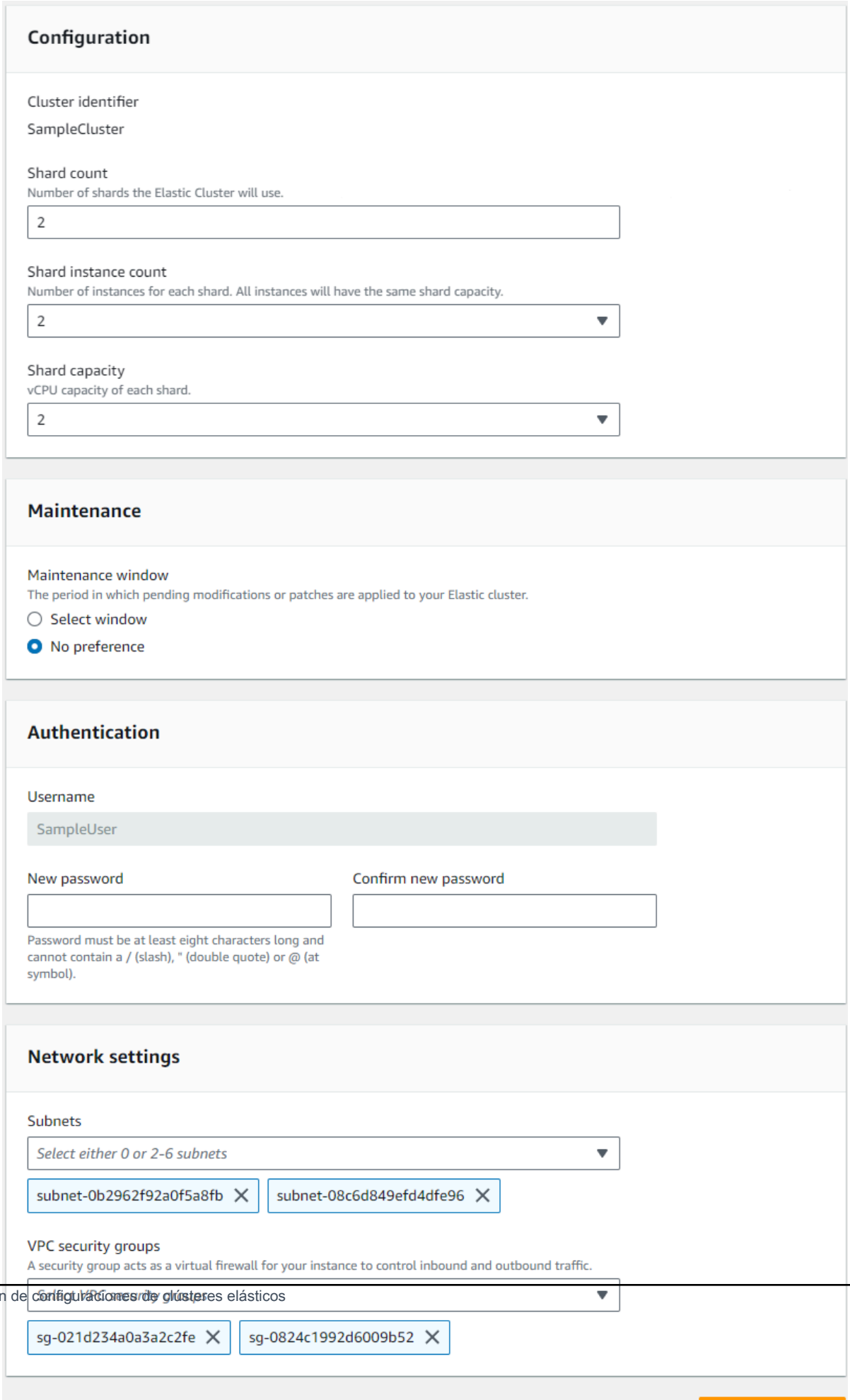
Using the Consola de administración de AWS

Para modificar la configuración de un clúster elástico mediante: Consola de administración de AWS

1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.
2. En el panel de navegación, seleccione Clusters (Clústeres).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú en la esquina superior izquierda del panel de navegación.

3. Elija el nombre del clúster que desea modificar en la columna Identificador de clúster.
4. Elija Modificar.
5. Edite los campos que desee cambiar y, a continuación, seleccione Modificar clúster.


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**
>
> Como alternativa, puede acceder al cuadro de diálogo Modificar clúster desde la página Clústeres si marca la casilla situada junto al clúster, selecciona Acciones y, a continuación, Modificar.

Using the AWS CLI

Para modificar la configuración de un clúster elástico mediante AWS CLI, utilice la `update-cluster` operación con los siguientes parámetros:

- **`--cluster-arn`**: obligatorio. El identificador del ARN del clúster que desee modificar.
- **`--shard-capacity`**: opcional. El número de vCPU asignado a cada partición. El máximo es 64. Los valores permitidos son 2, 4, 8, 16, 32, 64.
- **`--shard-count`**: opcional. La cantidad de particiones asignadas al clúster. El máximo es 32.
- Recuento de **`--shard-instance`**: opcional. El número de instancias de réplica que se aplican a todos los fragmentos de este clúster. El máximo es 16.
- **`--auth-type`**: opcional. El tipo de autenticación utilizado para determinar dónde buscar la contraseña que se usa para acceder al clúster elástico. Los tipos válidos son `PLAIN_TEXT` o `SECRET_ARN`.
- **`--admin-user-password`**: opcional. La contraseña asociada al usuario administrador.
- **`--vpc-security-group-ids`**: opcional. Configure una lista de grupos de seguridad de Amazon EC2 y la nube privada virtual (VPC) de Amazon para asociarlos a este clúster.
- **`--preferred-maintenance-window`**: opcional. El intervalo de tiempo semanal durante el cual puede llevarse a cabo el mantenimiento del sistema, en tiempo universal coordinado (UTC).

  El formato es: `ddd:hh24:mi-ddd:hh24:mi`. Días válidos: lunes, martes, miércoles, jueves, viernes, sábado, domingo

  El valor predeterminado es un periodo de 30 minutos seleccionado al azar de un bloque de 8 horas de tiempo para cada región de Amazon Web Services, que tiene lugar un día de la semana de forma aleatoria.

  El plazo mínimo es 30 minutos.
- **`--subnet-ids`**: opcional. Configure los identificadores de subred de la red.

Para utilizar el ejemplo siguiente, sustituya *user input placeholder* con su propia información.

Para 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
```

Para 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
```

Para supervisar el estado del clúster elástico después de la modificación, consulte Supervisión de un clúster elástico.

Monitoreo de un clúster elástico

En esta sección, explicamos cómo supervisar su clúster elástico mediante las instrucciones siguientes Consola de administración de AWS o siguiendo AWS CLI las instrucciones siguientes.

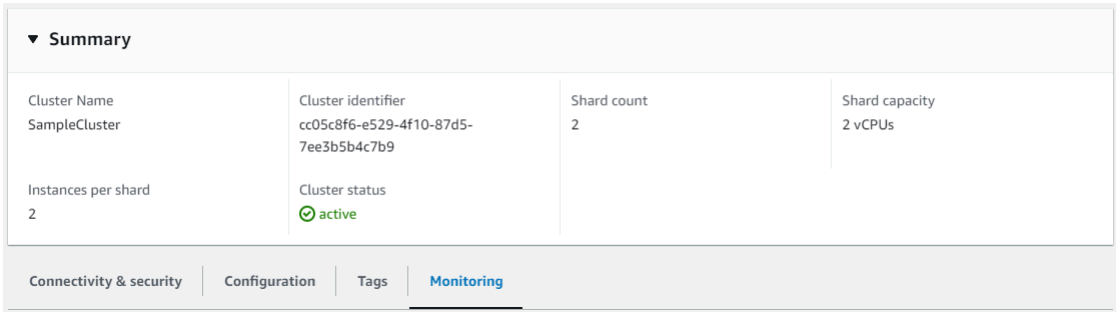
Using the Consola de administración de AWS

Para supervisar la configuración de un clúster elástico, utilice Consola de administración de AWS:

1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.
2. En el panel de navegación, seleccione Clusters (Clústeres).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú en la esquina superior izquierda del panel de navegación.

3. Elija el nombre del clúster que desea monitorear en la columna del Identificador del clúster.
4. Elija la pestaña Monitorización.



Se muestran varios gráficos de Amazon CloudWatch para las siguientes categorías de monitorización:

- Uso de los recursos
- Rendimiento
- Operaciones
- Sistema

También puedes acceder a Amazon a CloudWatch través del Consola de administración de AWS para configurar tu propio entorno de monitorización para tus clústeres elásticos.

Using the AWS CLI

Para supervisar una configuración específica de un clúster elástico mediante el AWS CLI, utilice la `get-cluster` operación con los siguientes parámetros:

- **`--cluster-arn`**: obligatorio. Identificador del ARN del clúster para el que desea obtener información detallada.

Para utilizar el ejemplo siguiente, sustituya *`user input placeholder`* con su propia información.

Para 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
```

Para Windows:

```
aws docdb-elastic get-cluster ^
    --cluster-arn arn:aws:docdb:-elastic:us-west-2:123456789012:cluster:/68ffcdf8-e3af-40a3-91e4-24736f2dacc9
```

La salida de esta operación será similar a lo que se indica a continuación:

```
"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",
        ...
 }
```

Para obtener más información, consulte `DescribeClusterSnapshot` en la Referencia de la API para administración de recursos de Amazon DocumentDB.

Para ver los detalles de todos los clústeres elásticos que utilizan el AWS CLI, utilice la `list-clusters` operación con los siguientes parámetros:

- **`--next-token`**: opcional. Si el número de elementos de salida (`--max-results`) es inferior al número total de elementos devueltos por las llamadas básicas a la API, la salida incluirá un `NextToken` que podrá pasar a un comando posterior para recuperar el siguiente conjunto de elementos.

- **`--max-results`**: opcional. El número total de elementos que se devuelven en la salida del comando. Si el número de registros es superior al valor `max-results` especificado, se incluye en la respuesta un token de paginación (`next-token`) que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.
  - Predeterminado: 100
  - Mínimo = 20, máximo = 100

Para utilizar el ejemplo siguiente, sustituya *`user input placeholder`* con su propia información.

Para Linux, macOS o Unix:

```
aws docdb-elastic list-clusters \
    --next-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ== \
    --max-results 2
```

Para Windows:

```
aws docdb-elastic list-clusters ^
    --next-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ== ^
    --max-results 2
```

La salida de esta operación será similar a lo que se indica a continuación:

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

Eliminación de un clúster elástico

En esta sección, explicamos cómo eliminar un clúster elástico mediante el comando Consola de administración de AWS o AWS CLI con las instrucciones siguientes.

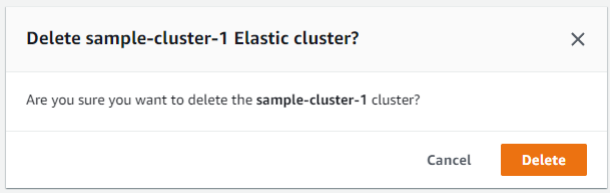
Using the Consola de administración de AWS

Para eliminar una configuración de clúster elástico mediante la Consola de administración de AWS:

1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.
2. En el panel de navegación, seleccione Clusters (Clústeres).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú en la esquina superior izquierda del panel de navegación.

3. En la tabla de lista de clúster, active la casilla de verificación situada a la izquierda del nombre del clúster que desea eliminar y, a continuación, elija Acciones. En el menú desplegable, seleccione Eliminar.
4. ¿En el clúster elástico Eliminar “cluster-name”? cuadro de diálogo, elija Eliminar.



La eliminación del clúster puede tardar varios minutos. Para monitorizar el estado del clúster, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

Using the AWS CLI

Para eliminar un clúster elástico mediante el AWS CLI, utilice la `delete-cluster` operación con los siguientes parámetros:

- **`--cluster-arn`**: obligatorio. El identificador del ARN del clúster que desee eliminar.
- **`--no-skip-final-backup`**: opcional. Si desea una instantánea final, debe incluir este parámetro con un nombre para la instantánea final. Debe incluir `--final-backup-identifier` o `--skip-final-backup`.
- **`--skip-final-backup`**: opcional. Use este parámetro solo si no desea realizar un respaldo final antes de eliminar el clúster. El comportamiento predeterminado es realizar una instantánea final.

En los siguientes ejemplos de AWS CLI código, se elimina un clúster con un ARN de arn:aws:docdb:us-west- 2:123456789012:sharded-cluster:sample-cluster con una copia de seguridad final.

En el siguiente ejemplo, sustituya cada una de ellas por su propia información. *`user input placeholder`*

Para 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
```

Para 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
```

En los siguientes ejemplos de AWS CLI código, se elimina un clúster con un ARN de arn:aws:docdb:us-west- 2:123456789012:sharded-cluster:sample-cluster sin realizar una copia de seguridad final.

Para utilizar el ejemplo siguiente, sustituya *user input placeholder* con su propia información.

Para 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 \
```

Para Windows:

```
aws docdb-elastic delete-cluster ^
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster ^
    --skip-final-backup ^
```

El resultado de la operación `delete-cluster` es el clúster que se va a eliminar.

La eliminación del clúster puede tardar varios minutos. Para monitorizar el estado del clúster, consulte Supervisión del estado de un clúster de Amazon DocumentDB.

Administración de instantáneas de los clústeres elásticos

Se pueden realizar instantáneas manuales después de crear un clúster elástico. Las copias de seguridad automatizadas se crean en el momento en que se crea la instantánea del clúster elástico.

> **Note**
>
> Su clúster elástico debe estar en estado `Available` para que se tome una instantánea manual.

En esta sección se explica cómo puede crear, ver, restaurar y eliminar instantáneas de clústeres elásticos.

En los temas siguientes se muestra cómo realizar diversas tareas al trabajar con instantáneas de clústeres elásticos de Amazon DocumentDB.

Temas

- Creación de una instantánea manual del clúster
- Visualización de una instantánea de un clúster elástico
- Restauración de un clúster desde una instantánea
- Copia de una instantánea de un clúster elástico
- Eliminar una instantánea de un clúster elástico
- Administración de una copia de seguridad automática de una instantánea de un clúster elástico

Creación de una instantánea manual del clúster

En esta sección, explicamos cómo crear una instantánea manual de un clúster elástico mediante las instrucciones siguientes o con ellas. Consola de administración de AWS AWS CLI

Using the Consola de administración de AWS

Para crear una instantánea manual de un clúster elástico mediante la Consola de administración de AWS:

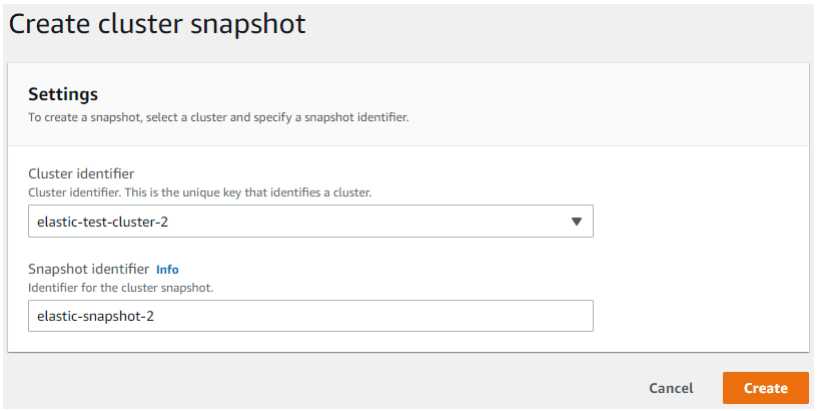
1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.
2. En el panel de navegación, elija Snapshots (Instantáneas).

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú en la esquina superior izquierda del panel de navegación.

3. En la página Snapshots (Instantáneas), elija Create (Crear).
4. En la página Crear una instantánea de clúster, en el campo Identificador del clúster, elija su clúster elástico en la lista desplegable.

   En el campo Identificador de instantáneas, introduce un identificador único para tu clúster elástico.

   Seleccione Crear.



> **Note**
>
> Como alternativa, puede acceder al cuadro de diálogo Crear una instantánea del clúster desde la página Clústeres, marcando la casilla situada junto al clúster, seleccionando Acciones y, a continuación, Realizar instantánea.

La instantánea de su clúster elástico se está aprovisionando. Esto puede tardar unos minutos en terminar. Puede verla y restaurarla desde su instantánea cuando el estado se muestre como `Available` en la lista de instantáneas.

Using the AWS CLI

Para crear una instantánea manual de un clúster elástico mediante el AWS CLI, utilice la `create-cluster-snapshot` operación con los siguientes parámetros:

- **`--snapshot-name`**: obligatorio. El nombre de la instantánea para crear un nuevo clúster.
- **`--cluster-arn`**: obligatorio. El identificador del ARN del clúster del que desea crear una instantánea.

Para utilizar el ejemplo siguiente, sustituya *`user input placeholder`* con su propia información.

Para 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
```

Para 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
```

Visualización de una instantánea de un clúster elástico

En esta sección, explicamos cómo ver la información de las instantáneas de un clúster elástico mediante las instrucciones siguientes Consola de administración de AWS o AWS CLI con ellas.

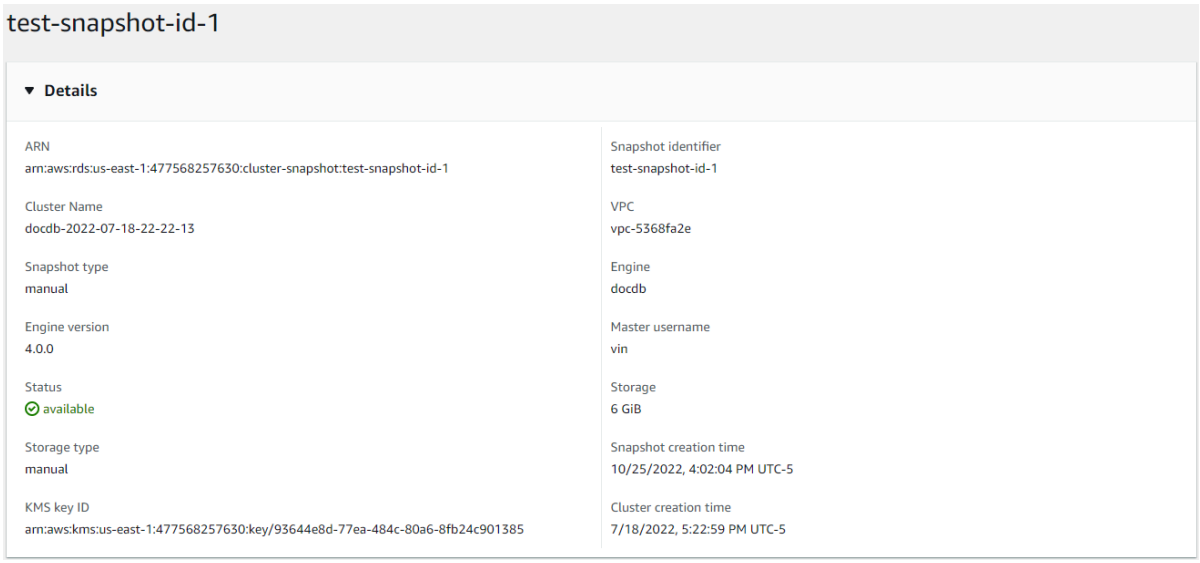
Using the Consola de administración de AWS

Para ver la información sobre una instantánea de un clúster elástico específica, utilice Consola de administración de AWS:

1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.
2. En el panel de navegación, elija Snapshots (Instantáneas).

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú en la esquina superior izquierda del panel de navegación.

3. En la página de instantáneas, elija su instantánea de la lista haciendo clic en el nombre en la columna de identificación de instantáneas.
4. Consulta la información de la instantánea en Detalles.



Using the AWS CLI

Para ver información sobre una instantánea específica de un clúster elástico mediante el AWS CLI, utilice la `get-cluster-snapshot` operación con los siguientes parámetros:

- **`--snapshot-arn`**: obligatorio. El identificador del ARN de la instantánea de la que desea obtener información.

Para utilizar el ejemplo siguiente, sustituya *`user input placeholder`* con su propia información.

Para Linux, macOS o Unix:

```
aws docdb-elastic get-cluster-snapshot \
    --snapshot-arn sampleResourceName
```

Para Windows:

```
aws docdb-elastic get-cluster-snapshot ^
    --snapshot-arn sampleResourceName
```

Para ver información sobre una instantánea de clúster elástico específica mediante la AWS CLI, utilice la `get-cluster-snapshot` operación con los siguientes parámetros:

- **`--snapshot-arn`**: obligatorio. El identificador del ARN de la instantánea de la que desea obtener información.

Para utilizar el ejemplo siguiente, sustituya *`user input placeholder`* con su propia información.

Para Linux, macOS o Unix:

```
aws docdb-elastic get-cluster-snapshot \
    --snapshot-arn sampleResourceName
```

Para Windows:

```
aws docdb-elastic get-cluster-snapshot ^
    --snapshot-arn sampleResourceName
```

Para ver información sobre todas las instantáneas del clúster elástico que utilizan el AWS CLI, utilice la `list-cluster-snapshots` operación con los siguientes parámetros:

- **`--snapshot-type`**: opcional. El tipo de instantáneas del clúster que se van a devolver. Puede especificar uno de los siguientes valores:
  - `automated`- Devuelve todas las instantáneas del clúster que Amazon DocumentDB ha creado automáticamente para su cuenta. AWS
  - `manual`- Devuelve todas las instantáneas del clúster que haya creado manualmente para su cuenta. AWS
  - `shared`- Devuelve todas las instantáneas manuales del clúster que se hayan compartido en su AWS cuenta.
  - `public`: se devuelven todas las instantáneas del clúster que se han marcado como públicas.
- **`--next-token`**: opcional. Un token de paginación opcional proporcionado por una solicitud anterior. Si se especifica este parámetro, la respuesta solo incluye registros más allá del token, hasta el valor especificado por `max-results`.
- **`--max-results`**: opcional. El número máximo de registros que se debe incluir en la respuesta. Si el número de registros es superior al valor `max-results` especificado, se incluye en la respuesta un token de paginación (`next-token`) que se conoce como marcador, de modo que se pueda recuperar el resto de resultados.

- Predeterminado: 100
- Mínimo = 20, máximo = 100

Para utilizar el ejemplo siguiente, sustituya *user input placeholder* con su propia información.

Para Linux, macOS o Unix:

```
aws docdb-elastic list-cluster-snapshots \
    --snapshot-type  value \
    --next-token  value \
    --max-results 50
```

Para Windows:

```
aws docdb-elastic list-cluster-snapshots ^
    --snapshot-type  value ^
    --next-token  value ^
    --max-results 50
```

Restauración de un clúster desde una instantánea

En esta sección, explicamos cómo restaurar un clúster elástico a partir de una instantánea, mediante las instrucciones siguientes Consola de administración de AWS o siguiendo AWS CLI las instrucciones siguientes.

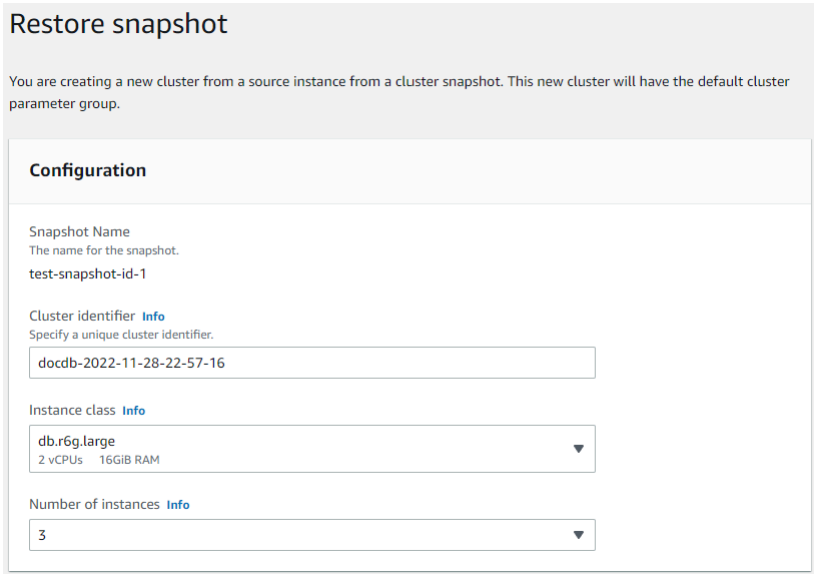
Using the Consola de administración de AWS

Para restaurar un clúster elástico a partir de una instantánea utilizando la Consola de administración de AWS:

1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.
2. En el panel de navegación, elija Snapshots (Instantáneas).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú en la esquina superior izquierda del panel de navegación.

3. Seleccione el botón situado a la izquierda de la instantánea que desea utilizar para restaurar un clúster, en la columna Identificador de instantánea.
4. En Acciones, elija Restaurar.



5. En la página Restaurar la instantánea, introduzca un nombre para el nuevo clúster en el campo identificador del clúster.

> **Note**
>
> Para cualquier restauración manual de instantáneas, debe crear un clúster nuevo.

6. En el campo Nube privada virtual (VPC), elija una VPC de la lista desplegable.
7. Para las subredes y los grupos de seguridad de VPC, puede usar los valores predeterminados o seleccionar tres subredes de su elección y hasta tres grupos de seguridad de VPC (uno como mínimo).
8. Si está satisfecho con la configuración del clúster, elija Restore cluster (Restaurar clúster) y espere mientras el clúster se restaura.

Using the AWS CLI

Para restaurar un clúster elástico a partir de una instantánea mediante el AWS CLI, utilice la `restore-cluster-from-snapshot` operación con los siguientes parámetros:

- **`--cluster-name`**: obligatorio. El nombre actual del clúster elástico tal como se ingresó durante la creación o se modificó por última vez.

- **`--snapshot-arn`**: obligatorio. El identificador del ARN de la instantánea que se utiliza para restaurar el clúster.
- **`--vpc-security-group-ids`**: opcional. Uno o más grupos de seguridad de Amazon EC2 y la nube privada virtual (VPC) de Amazon para asociar al clúster.
- **`--kms-key-id`**: opcional. Configure el identificador de la clave de KMS de un clúster cifrado.

  El identificador de la clave de KMS es el nombre de recurso de Amazon (ARN) de la clave de AWS KMS cifrado. Si está creando un clúster con la misma cuenta de Amazon Web Services a la que pertenece la clave de cifrado de KMS utilizada para cifrar el clúster nuevo, puede utilizar el alias de la clave de KMS en lugar del ARN para la clave de cifrado de KMS.

  Si no se especifica ninguna clave de cifrado KmsKeyId y el `StorageEncrypted` parámetro es verdadero, Amazon DocumentDB utiliza la clave de cifrado predeterminada.
- **`--subnet-ids`**: opcional. ID de subred de red.

En el siguiente ejemplo, sustituya cada una *`user input placeholder`* por su propia información.

Para 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
```

Para 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
```

Copia de una instantánea de un clúster elástico

En Amazon DocumentDB, puede copiar instantáneas manuales y automáticas de clústeres elásticos en la misma región y en la misma cuenta. En esta sección, explicamos cómo copiar una instantánea de un clúster elástico mediante el Consola de administración de AWS comando o. AWS CLI

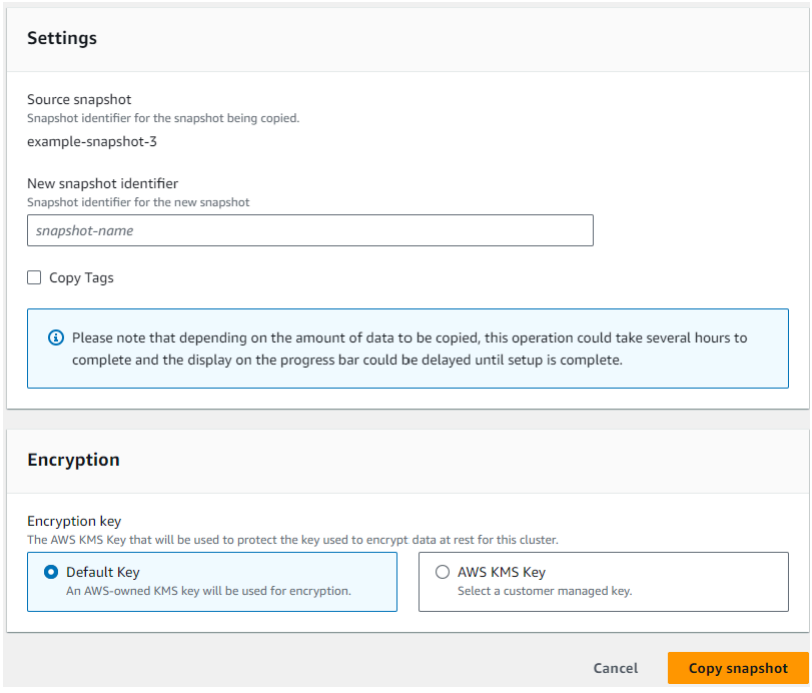
Using the Consola de administración de AWS

Para copiar una instantánea de un clúster elástico mediante Consola de administración de AWS:

1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.
2. En el panel de navegación, elija Snapshots (Instantáneas).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú en la esquina superior izquierda del panel de navegación.

3. Seleccione el botón situado a la izquierda de la instantánea que desea copiar en la columna Identificador de la instantánea.
4. Seleccione Acciones y, a continuación, Copiar.



5. En Identificador de la nueva instantánea, escriba un nombre para la instantánea nueva.

6. En Copiar etiquetas, active la casilla si desea copiar todas las etiquetas de la instantánea del clúster elástico de origen a la instantánea del clúster elástico de destino.
7. Para el cifrado, elija una clave de AWS KMS predeterminada o una clave de KMS de su elección. La segunda opción le permite seleccionar una clave de KMS existente que ya haya creado o crear una nueva.
8. Seleccione Copiar instantánea cuando haya terminado.

Using the AWS CLI

Para copiar una instantánea de un clúster elástico mediante AWS CLI, utilice la `copy-cluster-snapshot` operación con los siguientes parámetros:

- **`--source-db-cluster-snapshot-identifier`**: obligatorio. El identificador de la instantánea del clúster elástico existente que se está copiando. Debe existir la instantánea del clúster elástico y tener el estado Disponible. Este parámetro no distingue entre mayúsculas y minúsculas.
- **`--target-db-cluster-snapshot-identifier`**: obligatorio. El identificador de la nueva instantánea del clúster elástico que se va a crear a partir de la instantánea del clúster existente. Este parámetro no distingue entre mayúsculas y minúsculas.

  Restricciones relativas a la nomenclatura de instantáneas de destino:
  - No se puede coincidir con el nombre de una instantánea existente.
  - Debe tener [1-63] letras, números o guiones.
  - El primer carácter debe ser una letra.
  - No puede terminar con un guion ni contener dos guiones consecutivos.

Para utilizar el ejemplo siguiente, sustituya *`user input placeholder`* con su propia información.

Para Linux, macOS o Unix:

```
aws docdb-elastic copy-cluster-snapshot \
     --source-cluster-snapshot-arn <sample ARN> \
     --target-cluster-snapshot-name my-target-copied-snapshot
```

Para Windows:

```
aws docdb-elastic copy-cluster-snapshot ^
     --source-cluster-snapshot-arn <sample ARN> ^
     --target-cluster-snapshot-name my-target-copied-snapshot
```

Eliminar una instantánea de un clúster elástico

En esta sección, explicamos cómo eliminar una instantánea de un clúster elástico mediante el comando Consola de administración de AWS o AWS CLI.

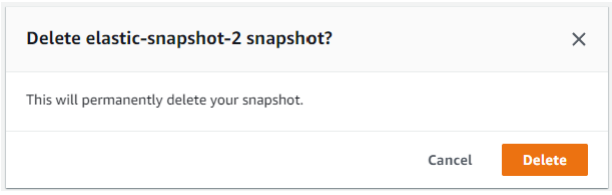
Using the Consola de administración de AWS

Para restaurar un clúster elástico a partir de una instantánea utilizando la Consola de administración de AWS:

1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.
2. En el panel de navegación, elija Snapshots (Instantáneas).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú en la esquina superior izquierda del panel de navegación.

3. Seleccione el botón situado a la izquierda de la instantánea que desea utilizar para restaurar un clúster, en la columna Identificador de instantánea.
4. Elija Acciones y, a continuación, elija Eliminar.



5. En el cuadro de diálogo Eliminar la instantánea “snapshot-name”, seleccione Eliminar.

Using the AWS CLI

Para eliminar una instantánea de un clúster elástico mediante el AWS CLI, utilice la `delete-cluster-snapshot` operación con los siguientes parámetros:

- **`--snapshot-arn`**: obligatorio. El identificador del ARN de la instantánea que se utiliza para restaurar el clúster.

En el siguiente ejemplo, sustituya cada uno *`user input placeholder`* por su propia información.

Para Linux, macOS o Unix:

```
aws docdb-elastic delete-cluster-snapshot \
     --snapshot-arn sampleResourceName
```

Para Windows:

```
aws docdb-elastic delete-cluster-snapshot ^
     --snapshot-arn sampleResourceName
```

Administración de una copia de seguridad automática de una instantánea de un clúster elástico

Amazon DocumentDB toma instantáneas diarias de los clústeres elásticos. Puede especificar la ventana de copia de seguridad preferida y el periodo de retención de la copia de seguridad en una configuración de instantáneas de clúster elástico nueva o existente. En esta sección, explicamos cómo configurar los parámetros de respaldo automáticos en una instantánea de clúster elástico, mediante el comando Consola de administración de AWS o AWS CLI.

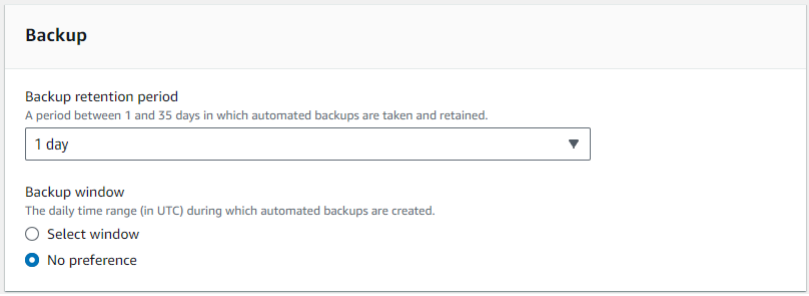
Using the Consola de administración de AWS

Para configurar una copia de seguridad automática para una nueva instantánea de un clúster elástico, utilice Consola de administración de AWS:

1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.
2. En el panel de navegación, seleccione Clusters (Clústeres).

> **Tip**
>
> Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú en la esquina superior izquierda del panel de navegación.

3. Seleccione el botón situado a la izquierda del clúster para el que desea cambiar la configuración de copia de seguridad en la columna Identificador del clúster.
4. Elija Acciones y, a continuación, Modificar.
5. En la sección Copia de seguridad, edite los campos de acuerdo con los requisitos de copia de seguridad.



   a. Periodo de retención de copias de seguridad: en la lista, elija el número de días que se deben conservar las copias de seguridad automáticas de este clúster antes de eliminarlas.
   b. Periodo de copia de seguridad: especifique la hora del día en que Amazon DocumentDB debe hacer las copias de seguridad de este clúster y la duración de las mismas.
      i. Elija Seleccionar periodo si desea configurar la hora y la duración de la creación de las copias de seguridad.

         Hora de inicio: en la primera lista, elija la hora de inicio (UTC) para las copias de seguridad automáticas. En la segunda lista, elija el minuto de la hora en que desea que comiencen las copias de seguridad automáticas.

         Duración: en la lista, elija el número de horas que se deben asignar para crear las copias de seguridad automáticas.
      ii. Seleccione Sin preferencia si desea que Amazon DocumentDB elija la hora y la duración de la creación de las copias de seguridad.
6. Elija Modificar clúster cuando haya terminado.

Using the AWS CLI

Para configurar una copia de seguridad automática para una nueva instantánea de un clúster elástico mediante el AWS CLI, utilice la `create-cluster-snapshot` operación con los siguientes parámetros:

- **`--preferred-backup-window`**: opcional. El intervalo de tiempo diario preferido durante el que se crean las copias de seguridad automatizadas. La ventana predeterminada es una ventana de 30 minutos seleccionada al azar de un bloque de 8 horas para cada Región de AWS una.

  Restricciones:

  - Tiene que tener el formato `hh24:mi-hh24:mi`.
  - Debe indicarse en Tiempo universal coordinado (UTC).
  - No debe entrar en conflicto con la ventana de mantenimiento preferida.
  - Debe durar al menos 30 minutos.
- **`--backup-retention-period`**: opcional. El número de días durante los que se retienen las copias de seguridad automatizadas. El valor predeterminado es 1.

  Restricciones:

  - Debe especificar un valor mínimo de 1.
  - El intervalo va de 1 a 35.

> **Note**
>
> Las copias de seguridad automatizadas solo se hacen cuando el clúster tenga el estado “activo”.

> **Note**
>
> También puede modificar los parámetros `preferred-backup-window` y `backup-retention-period` de un clúster elástico existente mediante el comando `aws docdb-elastic update-cluster`.

Para utilizar el ejemplo siguiente, sustituya *`user input placeholder`* con su propia información.

En el siguiente `create-cluster` ejemplo, se crea el clúster elástico de Amazon DocumentDB *`sample-cluster`* con un período de retención de *`7`* días para las copias de seguridad automáticas y una ventana de copia de seguridad preferida de*`18:00-18:30 UTC`*.

Para 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
```

Para 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
```

Detención e inicio de un clúster elástico de Amazon DocumentDB

La detención e inicio de los clústeres elásticos de Amazon DocumentDB puede ayudarlo a administrar los costos de entornos de desarrollo y pruebas. En lugar de crear y eliminar clústeres elásticos cada vez que utilice Amazon DocumentDB, puede detener temporalmente el clúster cuando ya no es necesario. A continuación, puede iniciarlo de nuevo cuando reanude sus pruebas.

Temas

- Información general de detención e inicio de un clúster elástico
- Operaciones que puede llevar a cabo en un clúster elástico detenido

Información general de detención e inicio de un clúster elástico

Durante los periodos en los que no necesite un clúster elástico de Amazon DocumentDB, puede detener el clúster. A continuación, puede volver a iniciar el clúster en cualquier momento que

necesite usarlo. El inicio y la detención simplifican los procesos de configuración y eliminación de clústeres elásticos usados para desarrollo, pruebas o actividades similares que no requieren disponibilidad continua. Puede detener e iniciar un clúster elástico mediante Consola de administración de AWS o AWS CLI con una sola acción.

Aunque el clúster elástico se detenga, el volumen de almacenamiento del clúster permanece sin cambios. Solo se le cobrará el almacenamiento, las instantáneas manuales y el almacenamiento de la copia de seguridad automática dentro de su intervalo de retención especificado. Después de siete días, Amazon DocumentDB inicia automáticamente el clúster elástico para que no pierda las actualizaciones de mantenimiento necesarias. Cuando el clúster se inicie después de siete días, se le comenzará a cobrar por el uso del clúster elástico de nuevo. Cuando el clúster se detenga, no podrá consultar el volumen de almacenamiento, ya que la consulta requiere que el clúster esté disponible.

Cuando se detiene un clúster elástico de Amazon DocumentDB, no se pueden modificar en modo alguno. Esto incluye la eliminación del clúster.

Using the Consola de administración de AWS

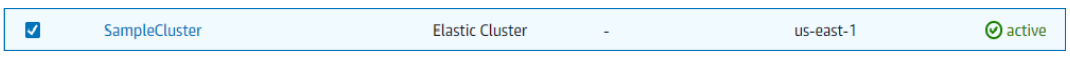
El procedimiento siguiente le muestra cómo detener un clúster elástico en un estado disponible o iniciar un clúster elástico detenido.

Detención o inicio de un clúster elástico de Amazon DocumentDB

1. Inicie sesión en y abra la Consola de administración de AWS consola de Amazon DocumentDB en https://console.aws.amazon.com/docdb.
2. En el panel de navegación, seleccione Clusters (Clústeres).

   > **Tip**
   >
   > Si no ve el panel de navegación del lado izquierdo de la pantalla, seleccione el icono de menú (≡) en la esquina superior izquierda de la página.

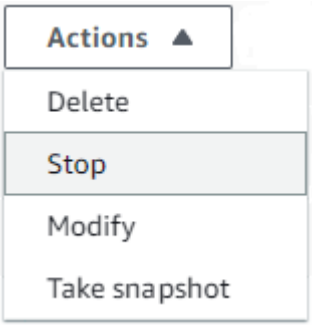
3. En la lista de clústeres, elija el botón situado a la izquierda del nombre del clúster que quiera detener o iniciar.

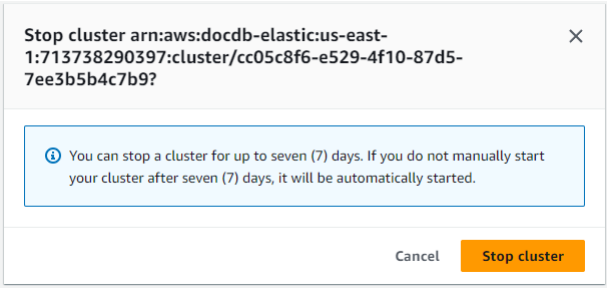
4. Elija Actions (Acciones) y, a continuación, elija la acción que desea realizar en el clúster.

- Si desea detener el clúster y el clúster está disponible:

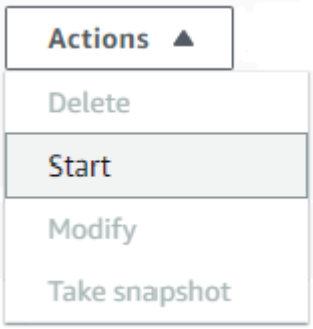
  a. Elija Detener.

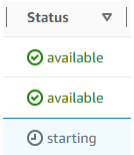
  b. En el cuadro de diálogo de confirmación, seleccione Detener clúster parar confirmar que desea detener el clúster elástico o elija Cancelar para mantener el clúster en ejecución.

- Si desea iniciar el clúster y este se detiene, seleccione Start (Iniciar).

5. Supervise el estado del clúster elástico. Si ha iniciado el clúster, podrá volver a utilizarlo cuando esté Disponible. Para obtener más información, consulte Determinación del estado de un clúster.

Using the AWS CLI

En los ejemplos de código siguientes se muestra cómo detener un clúster elástico en un estado activo o disponible o iniciar un clúster elástico detenido.

Para detener un clúster elástico que utilice el AWS CLI, utilice la `stop-cluster` operación. Para iniciar un clúster detenido, utilice la operación `start-cluster`. En ambas operaciones se utiliza el parámetro `--cluster-arn`.

Parámetro

- **`--cluster-arn`**: obligatorio. El identificador ARN del clúster elástico que desee detener o iniciar.

Example— Para detener un clúster elástico mediante la AWS CLI

Para utilizar el ejemplo siguiente, sustituya *`user input placeholder`* con su propia información.

El siguiente código detiene el clúster elástico con un ARN de `arn:aws:docdb-elastic:us-east-1:477568257630:cluster/b9f1d489-6c3e-4764-bb42-da62ceb7bda2`.

> **Note**
>
> El estado del clúster elástico debe ser activo o disponible.

Para 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
```

Para Windows:

```
aws docdb-elastic stop-cluster ^
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Example— Para iniciar un clúster elástico utilizando el AWS CLI

Para utilizar el ejemplo siguiente, sustituya *user input placeholder* con su propia información.

El siguiente código inicia el clúster elástico con un ARN de `arn:aws:docdb-elastic:us-east-1:477568257630:cluster/b9f1d489-6c3e-4764-bb42-da62ceb7bda2`.

> **Note**
>
> El clúster elástico debe estar detenido actualmente.

Para 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
```

Para Windows:

```
aws docdb-elastic start-cluster ^
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Operaciones que puede llevar a cabo en un clúster elástico detenido

No se puede modificar la configuración de un clúster elástico de Amazon DocumentDB mientras el clúster esté detenido. Debe iniciar el clúster antes de realizar cualquier tarea administrativa de ese tipo.

Amazon DocumentDB aplica cualquier mantenimiento programado a su clúster elástico detenido solo después de que se vuelva a iniciar. Después de siete días, Amazon DocumentDB inicia automáticamente un clúster elástico detenido para que no se quede demasiado rezagado en su estado de mantenimiento. Cuando el clúster elástico se reinicie, se le comenzará a cobrar por las particiones en el clúster de nuevo.

Cuando un clúster elástico está detenido, Amazon DocumentDB no hace copias de seguridad automatizadas ni amplía el periodo de retención de copia de seguridad.

Mantenimiento de clústeres elásticos de Amazon DocumentDB

Temas

- Visualización de las acciones de mantenimiento pendientes del clúster elástico
- Actualizaciones del motor de clústeres elásticos
- Actualizaciones del sistema operativo para los clústeres elásticos

Amazon DocumentDB realiza tareas de mantenimiento periódicas en los recursos del clúster elástico de Amazon DocumentDB. En la mayoría de los casos, el mantenimiento incluye actualizaciones del motor de base de datos (mantenimiento de clústeres elásticos) o del sistema operativo subyacente del clúster elástico (actualizaciones del sistema operativo). Las actualizaciones del motor de base de datos son parches necesarios e incluyen correcciones de seguridad, correcciones de errores y mejoras en el motor de base de datos. Si bien la mayoría de los parches del sistema operativo son opcionales, si no los aplica durante un tiempo, es posible que el parche sea necesario y se aplique automáticamente para mantener su postura de seguridad. Aplique las actualizaciones del sistema operativo a sus clústeres elásticos de Amazon DocumentDB tan pronto como estén disponibles.

Los parches del motor de base de datos requieren que desconecte los clústeres elásticos de Amazon DocumentDB durante un breve plazo de tiempo. Una vez disponibles, estos parches se programan automáticamente para que se apliquen durante un próximo período de mantenimiento programado del clúster elástico de Amazon DocumentDB.

Los clústeres elásticos tienen sus propios períodos de mantenimiento respectivos. Las modificaciones de los clústeres elásticos que haya decidido no aplicar inmediatamente también se aplican durante el período de mantenimiento. De forma predeterminada, cuando crea un clúster elástico, Amazon DocumentDB asigna un período de mantenimiento para el clúster elástico. Puede elegir el período de mantenimiento cuando crea el clúster elástico. También puede modificar los periodos de mantenimiento en cualquier momento para ajustarlos a las prácticas o las programaciones de su empresa. Por lo general, se recomienda elegir periodos de mantenimiento que minimicen la repercusión de las tareas de mantenimiento en la aplicación (por ejemplo, por las noches o durante los fines de semana).

Visualización de las acciones de mantenimiento pendientes del clúster elástico

Para ver si hay disponible una actualización de mantenimiento para un clúster elástico, use la AWS CLI.

Si hay disponible una actualización, puede realizar una de las acciones siguientes:

- Aplace una acción de mantenimiento que esté actualmente programada para el próximo periodo de mantenimiento (solo para los parches del sistema operativo).
- Aplicar inmediatamente las operaciones de mantenimiento.
- Programar las operaciones de mantenimiento para que se inicien durante el siguiente periodo de mantenimiento.
- Programe las acciones de mantenimiento para que se inicien durante el período de aplicación seleccionado.

El periodo de mantenimiento determina el momento en que comienzan las operaciones pendientes, pero no limita su tiempo total de ejecución.

Utilice la siguiente AWS CLI operación para determinar qué acciones de mantenimiento están pendientes. Enumere todas las acciones de mantenimiento pendientes:

```
aws docdb-elastic list-pending-maintenance-actions
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON):

```
{
'ResourcePendingMaintenanceActions': [
    {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'ENGINE_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'string',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
    },
],
'NextToken': 'string'
}
```

Obtenga la acción de mantenimiento pendiente (si la hubiera) en un `resourceArn` dado:

```
aws docdb-elastic get-pending-maintenance-action --resource-arn string-arn
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
{
    'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'ENGINE_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'string',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            }
        ]
    }
}
```

Parámetros:

- `ResourceArn`: el Nombre de recurso de Amazon (ARN) de Amazon DocumentDB del recurso al que se aplica la acción de mantenimiento pendiente.
- `Action`: la acción de mantenimiento pendiente que se aplica a este recurso.

  Valores válidos:

  - `ENGINE_UPDATE`
  - `ENGINE_UPGRADE`
  - `SECURITY_UPDATE`
  - `OS_UPDATE`
  - `MASTER_USER_PASSWORD_UPDATE`
- `AutoAppliedAfterDate`: primer período de mantenimiento después de esta fecha. `NEXT_MAINTENANCE OPT_IN` se ignora en este caso.
- `ForcedApplyDate`: se aplica independientemente del período de mantenimiento. `IMMEDIATE OPT_IN` se ignora en este caso.
- `OptInStatus`: un valor que indica el tipo de solicitud de alta o deshace una solicitud de alta. Una solicitud de alta de tipo `IMMEDIATE` no se puede deshacer.

Valores válidos:

- `IMMEDIATE`: aplicar inmediatamente la acción de mantenimiento.
- `NEXT_MAINTENANCE`: aplicar la acción de mantenimiento durante la siguiente ventana de mantenimiento del recurso.
- `APPLY_ON`: aplicar la acción de mantenimiento en la fecha de aplicación especificada, independientemente del siguiente período de mantenimiento del recurso.
- `UNDO_OPT_IN`: cancelar todas las solicitudes de alta de `NEXT_MAINTENANCE` o `APPLY_ON` existentes.

- `CurrentApplyDate`: se muestra si el tipo de suscripción es `APPLY_ON`.
- `Description`: una descripción de la opción para la acción de mantenimiento.

Actualizaciones del motor de clústeres elásticos

Con Amazon DocumentDB puede elegir el momento en que desea aplicar las operaciones de mantenimiento. Puede indicar cuándo Amazon DocumentDB debe aplicar las actualizaciones mediante la AWS CLI.

Aplique las acciones de mantenimiento pendientes:

```
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]
```

Parámetros:

- **`--resource-arn`**: el Nombre de recurso de Amazon (ARN) de Amazon DocumentDB del recurso al que se aplica la acción de mantenimiento pendiente.
- **`--apply-action`**: la acción de mantenimiento pendiente que se aplica a este recurso.

  Valores válidos:

  - `ENGINE_UPDATE`
  - `ENGINE_UPGRADE`
  - `SECURITY_UPDATE`
  - `OS_UPDATE`

  - MASTER_USER_PASSWORD_UPDATE
- **--opt-in-type**: un valor que indica el tipo de solicitud de alta o deshace una solicitud de alta. Una solicitud de alta de tipo IMMEDIATE no se puede deshacer.

  Valores válidos:

  - IMMEDIATE: aplicar inmediatamente la acción de mantenimiento.
  - NEXT_MAINTENANCE: aplicar la acción de mantenimiento durante la siguiente ventana de mantenimiento del recurso.
  - APPLY_ON: aplicar la acción de mantenimiento en la fecha de aplicación especificada, independientemente del siguiente período de mantenimiento del recurso.
  - UNDO_OPT_IN: cancelar todas las solicitudes de alta de NEXT_MAINTENANCE o APPLY_ON existentes.
- **[--apply-on]**: obligatorio si el tipo de suscripción es APPLY_ON. Formato: yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm (Esta opción usa la hora UTC. La hora de inicio puede ser en cualquier momento futuro, con un mínimo de 30 minutos y un máximo de 14 días, o una force/apply fecha en la que haya una acción pendiente, lo que ocurra primero. El intervalo de tiempo de principio a fin puede durar de un mínimo de 30 minutos a un máximo de 8 horas).

La salida de esta operación será similar a lo que se indica a continuación (formato JSON):

```
{
 'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'SECURITY_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'IMMEDIATE',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
 }
}
```

Parámetros:

- `ResourceArn`: el Nombre de recurso de Amazon (ARN) de Amazon DocumentDB del recurso al que se aplica la acción de mantenimiento pendiente.
- `Action`: la acción de mantenimiento pendiente que se aplica a este recurso.

  Valores válidos:

  - `ENGINE_UPDATE`
  - `ENGINE_UPGRADE`
  - `SECURITY_UPDATE`
  - `OS_UPDATE`
  - `MASTER_USER_PASSWORD_UPDATE`
- `AutoAppliedAfterDate`: primer período de mantenimiento después de esta fecha. `NEXT_MAINTENANCE OPT_IN` se ignora en este caso.
- `ForcedApplyDate`: se aplica independientemente del período de mantenimiento. `IMMEDIATE OPT_IN` se ignora en este caso.
- `OptInStatus`: un valor que indica el tipo de solicitud de alta o deshace una solicitud de alta. Una solicitud de alta de tipo `IMMEDIATE` no se puede deshacer.

  Valores válidos:

  - `IMMEDIATE`: aplicar inmediatamente la acción de mantenimiento.
  - `NEXT_MAINTENANCE`: aplicar la acción de mantenimiento durante la siguiente ventana de mantenimiento del recurso.
  - `APPLY_ON`: aplicar la acción de mantenimiento en la fecha de aplicación especificada, independientemente del siguiente período de mantenimiento del recurso.
  - `UNDO_OPT_IN`: cancelar todas las solicitudes de alta de `NEXT_MAINTENANCE` o `APPLY_ON` existentes.
- `CurrentApplyDate`: se muestra si el tipo de suscripción es `APPLY_ON`.
- `Description`: una descripción de la opción para la acción de mantenimiento.

Fechas de aplicación

Cada operación de mantenimiento tiene una fecha de aplicación que encontrará al describir las operaciones de mantenimiento pendientes. Al leer el resultado de las acciones de mantenimiento pendientes del AWS CLI, aparecen tres fechas:

- `CurrentApplyDate`: fecha en la que se aplicará la acción de mantenimiento inmediatamente o en la siguiente ventana de mantenimiento. Si el mantenimiento es opcional, este valor puede ser nulo.
- `ForcedApplyDate`: fecha en la que el mantenimiento se aplicará automáticamente, independientemente del período de mantenimiento.
- `AutoAppliedAfterDate`: fecha a partir de la cual se aplicará el mantenimiento durante el período de mantenimiento del clúster.

User-created acciones de mantenimiento

Como usuario de un clúster de Amazon DocumentDB, puede iniciar las actualizaciones de la configuración de los clústeres.

Actualización de la contraseña principal del clúster

```
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
```

Parámetros:

- **`--cluster-arn`**: el Nombre de recurso de Amazon (ARN) de Amazon DocumentDB del recurso al que se aplicará la acción de mantenimiento pendiente.
- **`[--admin-user-password]`**: la contraseña asociada al usuario administrador.
- **`[--auth-type]`**: el tipo de autenticación utilizado para determinar dónde buscar la contraseña que se usa para acceder al clúster elástico. Los tipos válidos son `PLAIN_TEXT` o `SECRET_ARN`.
- **`[--apply-method]`**: un valor que especifica el tipo de método que se aplica. Los valores permitidos son `IMMEDIATE` y `APPLY_ON`. El valor predeterminado es `IMMEDIATE`.
- **`[--apply-on]`**: obligatorio si `apply-method` es `APPLY_ON`. Formato: `yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm` (Esta opción usa la hora UTC. La hora de inicio puede ser en cualquier momento futuro, entre un mínimo de 30 minutos y un máximo de 14 días. El intervalo de tiempo de inicio a fin puede ser de un mínimo de 30 minutos a un máximo de 8 horas).

La salida de esta operación será similar a lo que se indica a continuación (formato JSON):

```
{
 'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'MASTER_USER_PASSWORD_UPDATE',
                'OptInStatus': 'APPLY_ON',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
 }
}
```

Cambio de los periodos de mantenimiento de Amazon DocumentDB

El periodo de mantenimiento debe corresponder al momento de mínimo uso y, por tanto, podría ser preciso modificarlo cada cierto tiempo. El clúster elástico deja de estar disponible durante este período solamente si se están aplicando cambios en el sistema (por ejemplo, se está realizando un cambio en la operación de escalado del almacenamiento) y se requiere una interrupción. Solamente dejará de estar disponible durante el tiempo mínimo requerido para realizar los cambios necesarios.

El valor predeterminado es un periodo de 30 minutos seleccionado al azar de un bloque de 8 horas de tiempo para cada región de Amazon Web Services, que tiene lugar un día de la semana de forma aleatoria.

Para cambiar el período de mantenimiento, consulte Modificación de configuraciones de clústeres elásticos.

Actualizaciones del sistema operativo para los clústeres elásticos

En ocasiones, los clústeres elásticos de Amazon DocumentDB requieren actualizaciones del sistema operativo. Amazon DocumentDB actualiza el sistema operativo a una versión más reciente para mejorar el rendimiento de la base de datos y la posición de seguridad general de los clientes. Las actualizaciones del sistema operativo no cambian la versión del motor del clúster elástico de Amazon DocumentDB.

La mayoría de las actualizaciones del sistema operativo para los clústeres elásticos de Amazon DocumentDB son opcionales y no tienen una fecha de aplicación fija. Sin embargo, si no aplica

estas actualizaciones durante un tiempo, es posible que acaben siendo necesarias y se apliquen automáticamente durante el período de mantenimiento de su clúster. Esto ayuda a mantener la posición de seguridad de la base de datos. Para evitar tiempos de inactividad inesperados, aplique las actualizaciones del sistema operativo a su clúster elástico de Amazon DocumentDB tan pronto como estén disponibles y configure el período de mantenimiento del clúster en un momento que se adapte a sus necesidades empresariales.

Activación del cifrado de datos en reposo de un clúster elástico de Amazon DocumentDB

Los siguientes temas le ayudan a conocer, crear y supervisar las claves de AWS Key Management Service cifrado para los clústeres elásticos de Amazon DocumentDB:

Temas

- Cómo utilizan los clústeres elásticos de Amazon DocumentDB las subvenciones en AWS KMS
- Creación de una clave administrada por el cliente
- Supervisión de las claves de cifrado de clústeres elásticos de Amazon DocumentDB
- Más información

Los clústeres elásticos de Amazon DocumentDB se integran automáticamente con AWS Key Management Service (AWS KMS) para administrar las claves y utilizan un método conocido como cifrado de sobres para proteger los datos. Para obtener más información acerca del cifrado de sobre, consulte Cifrado de sobre en la Guía para desarrolladores de AWS Key Management Service .

Un AWS KMS key es una representación lógica de una clave. La clave de KMS incluye metadatos, como el ID de clave, la fecha de creación, la descripción y el estado de la clave. La clave de KMS también contiene el material de claves utilizado para cifrar y descifrar datos. Para obtener más información acerca de las claves de KMS, consulte AWS KMS keys en la Guía para desarrolladores de AWS Key Management Service .

Los clústeres elásticos de Amazon DocumentDB admiten el cifrado con dos tipos de claves:

- AWS claves propias: los clústeres elásticos de Amazon DocumentDB utilizan estas claves de forma predeterminada para cifrar automáticamente los datos de identificación personal. No puede ver, administrar ni usar claves AWS propias ni auditar su uso. Sin embargo, no tiene que realizar ninguna acción ni cambiar ningún programa para proteger las claves que cifran sus datos. Para

obtener más información, consulte las claves propiedad de AWS en la Guía para desarrolladores de AWS Key Management Service .

- Customer-managed claves: son simétricas y tú AWS KMS keys las creas, posees y administras. Como usted tiene el control total de este cifrado, puede realizar dichas tareas como:
  - Establecer y mantener políticas de claves
  - Establecer y mantener concesiones y políticas de IAM
  - Habilitar y deshabilitar políticas de claves
  - Rotar el material criptográfico
  - Adición de etiquetas de
  - Crear alias de clave
  - Programar la eliminación de claves

  Para obtener más información, consulte las claves administradas por el cliente en la Guía para desarrolladores de AWS Key Management Service .

> **Important**
>
> Debe utilizar una clave de cifrado de KMS simétrica para cifrar el clúster, ya que Amazon DocumentDB solo admite claves de cifrado de KMS de cifrado simétricas. No utilice una CMK asimétrica para intentar cifrar los datos de los clústeres elásticos de Amazon DocumentDB. Para obtener más información, consulte claves asimétricas de AWS KMS en la Guía para desarrolladores de AWS Key Management Service .
> Si Amazon DocumentDB ya no puede obtener acceso a la clave de cifrado de un clúster, por ejemplo, cuando se revoca el acceso a una clave, el clúster cifrado entra en un estado terminal. En este caso, solo puede restaurar el clúster desde una copia de seguridad. Para Amazon DocumentDB, las copias de seguridad siempre están habilitadas durante 1 día. Además, si deshabilita la clave de un clúster cifrado de Amazon DocumentDB, eventualmente perderá el acceso de lectura y escritura a ese clúster. Cuando Amazon DocumentDB encuentra una instancia que está cifrada con una clave a la que no tiene acceso, pone el clúster en un estado terminal. En dicho estado, el clúster ya no está disponible y no es posible recuperar su estado actual. Para restaurar el clúster, debe volver a activar el acceso a la clave de cifrado para Amazon DocumentDB y después restaurar el clúster a partir de una copia de seguridad.

> **Important**
>
> No puede cambiar la clave KMS de un clúster cifrado después de haberlo creado. Asegúrese de determinar los requisitos de clave de cifrado antes de crear el clúster elástico cifrado.

Cómo utilizan los clústeres elásticos de Amazon DocumentDB las subvenciones en AWS KMS

Los clústeres elásticos de Amazon DocumentDB requieren una concesión para utilizar la clave administrada por el cliente.

Cuando crea un clúster cifrado con una clave gestionada por el cliente, los clústeres elásticos de Amazon DocumentDB crean una concesión en su nombre enviando una `CreateGrant` solicitud a. AWS KMS Las concesiones se AWS KMS utilizan para dar a los clústeres elásticos de Amazon DocumentDB acceso a una clave de KMS en una cuenta de cliente.

Los clústeres elásticos de Amazon DocumentDB necesitan la concesión para utilizar la clave administrada por el cliente para las siguientes operaciones internas:

- Envíe `DescribeKey` solicitudes AWS KMS para verificar que el ID de clave simétrico administrado por el cliente, introducido al crear un clúster elástico, sea válido.
- Envía `GenerateDataKey` solicitudes AWS KMS para generar claves de datos cifradas con tu clave gestionada por el cliente.
- Envíe `Decrypt` solicitudes AWS KMS para descifrar las claves de datos cifradas para que puedan usarse para cifrar sus datos.
- Puede revocar el acceso a la concesión o eliminar el acceso del servicio a la clave administrada por el cliente en cualquier momento. Si lo hace, los clústeres elásticos de Amazon DocumentDB no podrán acceder a ninguno de los datos cifrados por la clave administrada por el cliente, lo que afectará a las operaciones que dependen de esos datos.

Creación de una clave administrada por el cliente

Puede crear una clave simétrica administrada por el cliente mediante la API Consola de administración de AWS o la API. AWS KMS

Creación de claves simétricas administradas por el cliente

Siga los pasos para crear una clave simétrica administrada por el cliente en la Guía para AWS Key Management Service desarrolladores.

Política de claves

Las políticas de clave controlan el acceso a la clave administrada por el cliente. Cada clave administrada por el cliente debe tener exactamente una política de clave, que contiene instrucciones que determinan quién puede usar la clave y cómo puede utilizarla. Cuando crea la clave administrada por el cliente, puede especificar una política de clave. Para obtener más información, consulte la información de acceso a la clave de KMS que se encuentra en la descripción general de AWS Key Management Service de la Guía para desarrolladores de AWS Key Management Service .

Para utilizar la clave administrada por el cliente con los recursos de los clústeres elásticos de Amazon DocumentDB, se deben permitir las siguientes operaciones de API en la política de claves:

- `kms:CreateGrant`: agrega una concesión a una clave administrada por el cliente. Otorga el acceso de control a una clave de KMS específica, lo que permite el acceso a las operaciones de concesión que requieren los clústeres elásticos de Amazon DocumentDB. Para obtener más información sobre las concesiones, consulte Uso de concesiones AWS KMS en la Guía para desarrolladores de AWS Key Management Service .
- `kms:DescribeKey`— Proporciona los detalles de la clave administrada por el cliente para permitir que los clústeres elásticos de Amazon DocumentDB validen la clave.
- `kms:Decrypt`— Permite a los clústeres elásticos de Amazon DocumentDB utilizar la clave de datos cifrados almacenada para acceder a los datos cifrados.
- `kms:GenerateDataKey`— Permite que los clústeres elásticos de Amazon DocumentDB generen una clave de datos cifrada y la almacenen, ya que la clave de datos no se utiliza inmediatamente para cifrar.

Para obtener más información, consulte los permisos para los AWS servicios en las políticas de claves y la solución de problemas de acceso a las claves en la Guía para AWS Key Management Service desarrolladores.

Restricción del acceso a las claves gestionadas por el cliente mediante políticas de IAM

Además de las políticas clave de KMS, también puede restringir los permisos de clave de KMS en una política de IAM.

Puede hacer que la política de IAM sea más estricta de varias maneras. Por ejemplo, para permitir que la clave administrada por el cliente se utilice solo para solicitudes que se originen en clústeres

elásticos de Amazon DocumentDB, puede utilizar la `kms:ViaService`clave de condición con el valor `docdb-elastic.<region-name>.amazonaws.com`.

Para obtener más información, consulte Cómo permitir a los usuarios de otras cuentas utilizar una clave de KMS en la Guía para desarrolladores de AWS Key Management Service .

Supervisión de las claves de cifrado de clústeres elásticos de Amazon DocumentDB

Cuando utiliza una clave gestionada por el cliente con sus recursos de clústeres elásticos de Amazon DocumentDB, puede utilizar AWS CloudTrail Amazon CloudWatch Logs para realizar un seguimiento de las solicitudes a las que envían los clústeres elásticos de Amazon DocumentDB. AWS KMS

Los siguientes ejemplos son AWS CloudTrail eventos para`CreateGrant`, `GenerateDataKeyWithoutPlainTextDecrypt`, y `DescribeKey` para supervisar AWS KMS key las operaciones convocadas por los clústeres elásticos de Amazon DocumentDB para acceder a los datos cifrados por la clave gestionada por el 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"
}
```


Más información

Los siguientes recursos proporcionan más información sobre cifrado de datos en reposo:

- Para obtener más información sobre AWS KMS los conceptos, consulte los conceptos AWS Key Management Service básicos en la Guía para AWS Key Management Service desarrolladores.

- Para obtener más información sobre AWS KMS la seguridad, consulte las prácticas recomendadas de seguridad AWS Key Management Service de la Guía para AWS Key Management Service desarrolladores.

Service-linked funciones en los clústeres elásticos

Los clústeres elásticos de Amazon DocumentDB utilizan AWS Identity and Access Management funciones vinculadas a servicios (IAM). Un rol vinculado a un servicio es un tipo único de rol de IAM que está vinculado directamente a los clústeres elásticos de Amazon DocumentDB. Service-linked los roles están predefinidos por los clústeres elásticos de Amazon DocumentDB e incluyen todos los permisos que el servicio necesita para llamar a otros AWS servicios en su nombre.

Un rol vinculado a servicios simplifica la configuración de Amazon DocumentDB porque ya no tendrá que agregar manualmente los permisos necesarios. Los clústeres elásticos de Amazon DocumentDB definen los permisos de sus roles vinculados al servicio y, a menos que esté definido de otra manera, solo los clústeres elásticos de Amazon DocumentDB pueden asumir sus roles. Los permisos definidos incluyen las políticas de confianza y de permisos, y que la política de permisos no se pueda asociar a ninguna otra entidad de IAM. Las funciones se pueden eliminar únicamente después de eliminar primero sus recursos relacionados. De esta forma, se protegen los recursos de los clústeres elásticos de Amazon DocumentDB, ya que se evita que se puedan eliminar accidentalmente permisos de acceso a los recursos.

Para obtener información sobre otros servicios que admiten funciones vinculadas a servicios, consulte AWS los servicios que funcionan con IAM y busque los servicios que están marcados con Sí en la columna Función. Service-Linked Elija una opción Sí con un enlace para ver la documentación acerca del rol vinculado al servicio en cuestión.

Service-linked permisos de rol para clústeres elásticos

Los clústeres elásticos de Amazon DocumentDB utilizan la función vinculada a servicios denominada `AWS ServiceRoleForDocDB-Elastic` para permitir que los clústeres elásticos de Amazon DocumentDB llamen a los AWS servicios en nombre de sus clústeres.

Este rol vinculado al servicio tiene una política de permisos adjunta llamada `AmazonDocDB-ElasticServiceRolePolicy` que le otorga permisos para operar en su cuenta. La política de permisos del rol permite que los clústeres elásticos de Amazon DocumentDB realicen las siguientes acciones en los recursos especificados:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/DocDB-Elastic"
                    ]
                }
            }
        }
    ]
}
```

> **Note**
>
> Debe configurar permisos para permitir a una entidad de IAM (como un usuario, grupo o rol) crear, editar o eliminar un rol vinculado a servicios. Si aparece el siguiente mensaje de error: “No se puede crear el recurso. Verifique que tiene permiso para crear el rol vinculado al servicio. De lo contrario, espere y vuelva a intentarlo más tarde”., asegúrese de tener habilitados los siguientes permisos:

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

Para obtener más información, consulte los permisos de Service-linked rol en la Guía del usuario de IAM.

Creación de un rol vinculado a servicio para los clústeres elásticos de Amazon DocumentDB

No necesita crear manualmente un rol vinculado a servicios. Cuando crea una instancia de base de datos, los clústeres elásticos de Amazon DocumentDB vuelven a crear por usted el rol vinculado al servicio.

Edición de un rol vinculado a servicio para los clústeres elásticos de Amazon DocumentDB

Los clústeres elásticos de Amazon DocumentDB no permiten editar el rol vinculado a servicios de `AWS ServiceRoleForDocDB-Elastic`. Después de crear un rol vinculado al servicio, no podrá cambiar el nombre del rol, ya que varias entidades podrían hacer referencia al rol. Sin embargo, sí puede editar la descripción del rol con IAM. Para obtener más información, consulte Editar un rol vinculado a servicios en la Guía del usuario de IAM.

Eliminación de un rol vinculado a servicio para los clústeres elásticos de Amazon DocumentDB

Si ya no necesita usar una función o un servicio que requiera un rol vinculado a un servicio, elimínelo. De esta forma no tiene una entidad no utilizada que no se monitoree ni mantenga de forma activa. Sin embargo, debe eliminar todos los clústeres para poder eliminar el rol vinculado al servicio.

Limpiar un rol vinculado a servicios

Antes de poder utilizar IAM para eliminar un rol vinculado a un servicio, primero debe confirmar que dicho rol no tiene sesiones activas y eliminar los recursos que utiliza.

Para comprobar si el rol vinculado a un servicio tiene una sesión activa en la consola de IAM:

1. Inicie sesión en la Consola de administración de AWS y abra la consola de IAM.

2. En el panel de navegación de la consola de IAM, elija Roles (Roles). A continuación, seleccione el nombre (no la casilla de verificación) del rol de `AWS ServiceRoleForDocDB-Elastic`.
3. En la página Summary (Resumen) del rol seleccionado, elija la pestaña Access Advisor (Acceso a Advisor).

> **Note**
>
> Si no está seguro de si los clústeres elásticos de Amazon DocumentDB utilizan el rol `AWS ServiceRoleForDocDB-Elastic`, puede intentar eliminar el rol para comprobarlo. Si el servicio usa el rol, se produce un error al eliminarlo y puede ver Regiones de AWS dónde se usa el rol. Si el rol se está utilizando, debe esperar que la sesión finalice para poder eliminarlo. No se puede revocar la sesión de un rol vinculado a servicios.
> Si desea eliminar el rol `AWS ServiceRoleForDocDB-Elastic`, primero debe eliminar todos sus clústeres.

Eliminación de todos los clústeres

Para eliminar un clúster en la consola de Amazon DocumentDB:

1. Inicie sesión en la Consola de administración de AWS de Amazon DocumentDB y ábrala.
2. En el panel de navegación, seleccione Clusters (Clústeres).
3. Elija el clúster que desea eliminar.
4. En Actions (Acciones), seleccione Delete (Eliminar).
5. Si aparece el mensaje ¿Crear instantánea final?, elija Sí o No.
6. Si eligió Yes (Sí) en el paso anterior, en Final snapshot name (Nombre de instantánea final) escriba el nombre de la instantánea final.
7. Elija Eliminar.

> **Note**
>
> Puede utilizar la consola de IAM, la CLI de IAM o la API de IAM para eliminar el rol vinculado a servicios `AWS ServiceRoleForDocDB-Elastic`. Para obtener más información, consulte Eliminación de un rol vinculado a un servicio en la Guía del usuario de IAM.

Cuotas de Amazon DocumentDB

En este tema se describen las cuotas de recursos y las restricciones de nomenclatura de Amazon DocumentDB (compatibles con MongoDB).

Para determinadas funciones de administración, Amazon DocumentDB utiliza tecnología operativa que se comparte con Amazon Relational Database Service (Amazon RDS) y Amazon Neptune.

Temas

- Tipos de instancias admitidas
- Regiones admitidas
- Cuotas regionales
- Cuotas de agregación
- Cuotas de clúster
- Cuotas de instancias
- Restricciones en la nomenclatura
- Restricciones de TTL
- Cuotas de clúster elásticas
- Cuotas de particiones de clústeres elásticos
- Cuotas de CPU, memoria, conexión y cursor de los clústeres elásticos por fragmento

Tipos de instancias admitidas

Amazon DocumentDB admite instancias bajo demanda y los siguientes tipos de instancias:

- NVMe-backed:
  - Tipos de instancias R6GD: `db.r6gd.xlarge`, `db.r6gd.2xlarge`, `db.r6gd.4xlarge`, `db.r6gd.8xlarge`, `db.r6gd.12xlarge`, `db.r6gd.16xlarge`.
- Optimizada para memoria:
  - Tipos de instancias 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`

  - Tipos de instancias R6G: `db.r6g.large`,,,,`db.r6g.xlarge`,`db.r6g.2xlarge`,`db.r6g.4xlarge`. `db.r6g.8xlarge` `db.r6g.12xlarge` `db.r6g.16xlarge`
  - Tipos de instancias R5: `db.r5.large`,,,`db.r5.xlarge`,`db.r5.2xlarge`,`db.r5.4xlarge`,`db.r5.8xlarge`. `db.r5.12xlarge` `db.r5.16xlarge`
  - Tipos de instancias de R4: `db.r4.large`, `db.r4.2xlarge`, `db.r4.4xlarge`, `db.r4.8xlarge`, `db.r4.16xlarge`.
- Rendimiento ampliable:
  - Tipos de instancias T4G: `db.t4g.medium`.
  - Tipo de instancias T3: `db.t3.medium`.

Para obtener más información sobre los tipos de instancias admitidos y sus especificaciones, consulte Especificaciones de clases de instancias.

Regiones admitidas

Amazon DocumentDB está disponible en los siguientes lugares: Regiones de AWS

Nombre de la región	Region	Zonas de disponibilidad (cálculo)
Este de EE. UU. (Ohio)	`us-east-2`	3
Este de EE. UU. (Norte de Virginia)	`us-east-1`	6
Oeste de EE. UU. (Oregón)	`us-west-2`	4
África (Ciudad del Cabo)	`af-south-1`	3
América del Sur (São Paulo)	`sa-east-1`	3
Asia-Pacífico (Hong Kong)	`ap-east-1`	3
Asia-Pacífico (Hyderabad)	`ap-south-2`	3
Asia-Pacífico (Malasia)	`ap-southeast-5`	3
Asia-Pacífico (Mumbai)	`ap-south-1`	3
Asia-Pacífico (Osaka)	`ap-northeast-3`	3
Asia-Pacífico (Seúl)	`ap-northeast-2`	4
Asia-Pacífico (Singapur)	`ap-southeast-1`	3

Nombre de la región	Region	Zonas de disponibilidad (cálculo)
Asia-Pacífico (Sídney)	ap-southeast-2	3
Asia-Pacífico (Yakarta)	ap-southeast-3	3
Asia-Pacífico (Melbourne)	ap-southeast-4	3
Asia-Pacífico (Tailandia)	ap-southeast-7	3
Asia-Pacífico (Tokio)	ap-northeast-1	3
Canadá (centro)	ca-central-1	3
Oeste de Canadá (Calgary)	ca-west-1	3
Región China (Pekín)	cn-north-1	3
China (Ningxia)	cn-northwest-1	3
Europa (Fráncfort)	eu-central-1	3
Europa (Zúrich)	eu-central-2	3
Europa (Irlanda)	eu-west-1	3
Europa (Londres)	eu-west-2	3
Europa (Milán)	eu-south-1	3
Europa (París)	eu-west-3	3
Europa (España)	eu-south-2	3
Europa (Estocolmo)	eu-north-1	3
México (centro)	mx-central-1	3

Nombre de la región	Region	Zonas de disponibilidad (cálculo)
Medio Oriente (EAU)	`me-central-1`	3
Israel (Tel Aviv)	`il-central-1`	3
AWS GovCloud (US-West)	`us-gov-west-1`	3
AWS GovCloud (US-East)	`us-gov-east-1`	3

Cuotas regionales

Para determinadas funciones de administración, Amazon DocumentDB utiliza tecnología operativa que se comparte con Amazon Relational Database Service (Amazon RDS). La siguiente tabla contiene las cuotas regionales que se comparten entre Amazon DocumentDB y Amazon RDS.

> **Note**
>
> La tecnología compartida de Amazon RDS descrita anteriormente solo se aplica a los clústeres basados en instancias de Amazon DocumentDB. Los clústeres elásticos de Amazon DocumentDB no comparten tecnología con Amazon RDS.

Las siguientes cuotas se aplican a los clústeres basados en instancias de Amazon DocumentDB y son por región. Cuenta de AWS

Recurso	AWS cuota predeterminada
Clústeres	40
Grupos de parámetros de clústeres	50
Clústeres por grupo de parámetros	100
Instancias por grupo de parámetros	300

Recurso	AWS cuota predeterminada
Suscripciones de eventos	20
Instancias	40
Instantáneas de clústeres manuales	100
Réplicas de lectura por clúster	15
Grupos de subredes	50
Subredes por grupo de subredes	20
Etiquetas por recurso	50
Grupos de seguridad de VPC por instancia	5

Las siguientes cuotas se aplican a los clústeres elásticos de Amazon DocumentDB y son Cuenta de AWS por región.

Recurso	AWS cuota predeterminada
Clústeres elásticos	20
vCPU de clústeres elásticos	1024
Instantánea de un clúster elástico manual	20

Puede utilizar Service Quotas para solicitar un aumento de una cuota, si esta es ajustable. Algunas solicitudes se resuelven automáticamente, mientras que otras se envían a Soporte. Puedes hacer un seguimiento del estado de una solicitud de aumento de cuota que se haya enviado a Soporte. Las solicitudes para aumentar las Service Quotas no reciben soporte prioritario. Si tienes una solicitud urgente, ponte en contacto con Soporte. Para obtener más información acerca de las Service Quotas, consulte What Is Service Quotas?

Para solicitar un aumento de cuota de Amazon DocumentDB:

1. Abra la consola de Service Quotas en https://console.aws.amazon.com/servicequotas y, si es necesario, inicie sesión.
2. En el panel de navegación, elija Servicios de AWS .
3. Seleccione clúster elástico de Amazon DocumentDB o Amazon DocumentDB (compatible con MongoDB) en la lista o escriba cualquiera de estas dos opciones en el campo de búsqueda.
4. Si la cuota es ajustable, puede seleccionar su botón de opción o su nombre y, a continuación, seleccionar Request quota increase (Solicitar aumento de cuota) en la parte superior derecha de la página.
5. En Change quota value (Cambiar valor de cuota), introduzca el nuevo valor. El nuevo valor debe ser mayor que el valor actual.
6. Seleccione Solicitar. Una vez resuelta la solicitud, Valor de cuota aplicada para la cuota se establece en el nuevo valor.
7. Para ver las solicitudes pendientes o resueltas recientemente, elija Panel en el panel de navegación. Para las solicitudes pendientes, seleccione el estado de la solicitud para abrir la recepción de solicitud. El estado inicial de una solicitud es `Pending`. Cuando el estado cambie a`Quota requested`, verás el número de caso con Soporte. Elija el número de caso para abrir el ticket para su solicitud.

Cuotas de agregación

En la siguiente tabla se describen las cuotas de agregación en Amazon DocumentDB.

Recurso	Cuota
Número máximo de etapas admitidas	500

Cuotas de clúster

La siguiente tabla describe las cuotas de clústeres basadas en instancias de Amazon DocumentDB.

Recurso	Cuota
Tamaño del clúster (suma de todas las colecciones e índices)	256 TiB para la versión 8.0 del motor y posteriores (128 TiB para las versiones anteriores del motor)
Tamaño de la colección (la suma de todas las colecciones no puede superar la cuota del clúster): no incluye el tamaño del índice	32 TiB
Colecciones por clúster	100 000
Bases de datos por clúster	100 000
Tamaño de la base de datos (la suma de todas las bases de datos no puede superar la cuota del clúster)	256 TiB para la versión 8.0 del motor y posteriores (128 TiB para las versiones anteriores del motor)
Profundidad de anidamiento de documentos	200 niveles
Tamaño del documento	16 MiB
Tamaño de clave de índice	2048 bytes

Recurso	Cuota
Índices por colección	64
Claves de un índice compuesto	32
Número máximo de operaciones de escritura en un único comando por lotes	100 000
Número de usuarios por clúster	1 000

Cuotas de instancias

En la siguiente tabla se describen las cuotas de Amazon DocumentDB por instancia.

Amazon DocumentDB 8.0

Tipo de instancia	Memoria de la instancia (GiB)	Conexiones (todas)	Cuota de cursores	Transacciones abiertas	Conexiones (activas)
T3.medium	4	1 000	30	50	102
T4G.medium	4	1 000	30	50	102
R5.large	16	3400	450	200	1 100
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	1 100
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 de instancia	Memoria de la instancia (GiB)	Conexiones (todas)	Cuota de cursores	Transacciones abiertas	Conexiones (activas)
R6G.12xlarge*	384	60000	2280	4800	4500
R6G.16xlarge*	512	60000	3040	6400	4500
R8G.large	16	3400	450	200	1 100
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

* Incluido R6GD.

Amazon DocumentDB 5.0

Tipo de instancia	Memoria de la instancia (GiB)	Conexiones (todas)	Cuota de cursores	Transacciones abiertas	Conexiones (activas)
T3.medium	4	1 000	30	50	102
T4G.medium	4	1 000	30	50	102
R5.large	16	3400	450	200	1 100
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	1 100
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

Tipo de instancia	Memoria de la instancia (GiB)	Conexiones (todas)	Cuota de cursores	Transacciones abiertas	Conexiones (activas)
R6G.16xlarge*	512	60000	3040	6400	4500
R8G.large	16	3400	450	200	1 100
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

* Incluido R6GD.

Amazon DocumentDB 4.0

Tipo de instancia	Memoria de la instancia (GiB)	Conexiones (todas)	Cuota de cursores	Transacciones abiertas	Conexiones (activas)
T3.medium	4	1 000	30	50	102

Tipo de instancia	Memoria de la instancia (GiB)	Conexiones (todas)	Cuota de cursores	Transacciones abiertas	Conexiones (activas)
T4G.medium	4	1 000	30	50	102
R5.large	16	3400	450	200	1 100
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	1 100
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

* Incluido R6GD.

Amazon DocumentDB 3.6

Tipo de instancia	Memoria de la instancia (GiB)	Conexiones (todas)	Cuota de cursores	Transacciones abiertas	Conexiones (activas)
T3.medium	4	500	30	N/A	102
R4.large	15.25	1700	450	N/A	1 100
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	1 100
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

Puedes supervisar y emitir alarmas sobre los límites por instancia mediante las siguientes CloudWatch métricas. Para obtener más información sobre las CloudWatch métricas de Amazon DocumentDB, consulteSupervisión de Amazon DocumentDB con CloudWatch.

Recurso	CloudWatch métrica de cuota	CloudWatch métrica de uso (1 minuto como máximo)	CloudWatch métrica de uso
Memoria de la instancia	-	-	FreeableMemory
Conexiones (todas)	DatabaseConnectionsLimit	DatabaseConnectionsMax	DatabaseConnections
Cursores	DatabaseCursorsLimit	DatabaseCursorsMax	DatabaseCursors
Transacciones	TransactionsOpenLimit	TransactionsOpenMax	TransactionsOpen

Restricciones en la nomenclatura

En la siguiente tabla se describen las restricciones de la nomenclatura en Amazon DocumentDB.

Recurso	Cuota predeterminada
Identificador de clúster	• Debe tener [1-63] letras, números o guiones. • El primer carácter debe ser una letra. • No puede terminar con un guion ni contener dos guiones consecutivos. • Debe ser único para todos los clústeres (en Amazon RDS, Amazon Neptune y Amazon DocumentDB) por AWS cuenta y región.
Identificador de instancia	• Debe tener [1-63] letras, números o guiones • El primer carácter debe ser una letra • No puede terminar por un guion ni contener dos guiones consecutivos • Debe ser único para todas las instancias (en Amazon RDS, Amazon Neptune y Amazon DocumentDB) por cuenta y región. AWS
Nombre de la colección: `<col>`	La longitud es de [1 a 255] caracteres (Amazon DocumentDB 5.0 y versiones posteriores).

Recurso	Cuota predeterminada
	La longitud es de [1 a 57] caracteres (Amazon DocumentDB 4.0 y versiones anteriores).
Nombre de la base de datos: <db>	La longitud es [1-63] caracteres.
Nombre de colección totalmente cualificado: <db>.<col>	La longitud es de [3 a 255] caracteres (Amazon DocumentDB 5.0 y versiones posteriores). La longitud es de [3—120] caracteres (Amazon DocumentDB 4.0 y versiones anteriores).
Nombre de índice completo: <db>.<col>$<index>	La longitud es de [6—511] caracteres (Amazon DocumentDB 5.0 y versiones posteriores). La longitud es de [6—127] caracteres (Amazon DocumentDB 4.0 y versiones anteriores).
Nombre de índice	La longitud es de [1 a 255] caracteres (Amazon DocumentDB 5.0 y versiones posteriores). La longitud es de [3 a 63] caracteres para el conjunto <col>$<index> (Amazon DocumentDB 4.0 y versiones anteriores).

Recurso	Cuota predeterminada

Contraseña principal	• La longitud es de [8-100] caracteres ASCII imprimibles.
	• Se puede utilizar cualquier carácter ASCII imprimible, excepto los siguientes:
	  • **/** (barra inclinada)
	  • **"** (comillas dobles)
	  • **@** (símbolo de arroba)

Nombre de usuario principal	• La longitud es de [1-63] caracteres alfanuméricos. • El primer carácter debe ser una letra. • No puede ser una palabra reservada por el motor de base de datos.
Nombre del grupo de parámetros	• La longitud es de [1-255] caracteres alfanuméricos. • El primer carácter debe ser una letra. • No puede terminar con un guion ni contener dos guiones consecutivos.

Restricciones de TTL

Las eliminaciones de un índice de TTL no están garantizadas en un periodo de tiempo específico y se efectúan en la medida que sea posible. Factores como la utilización de recursos de instancia, el tamaño del documento y el rendimiento general pueden afectar la temporización de una eliminación de TTL.

Cuotas de clúster elásticas

En la siguiente tabla se describen las cuotas máximas para los clústeres elásticos de Amazon DocumentDB.

Recurso	Cuota
Clústeres elásticos por región	20
La suma de vCPU en todos los clústeres elásticos por región	1024
Instantáneas manuales de clústeres por región	20
Particiones por clúster	32
Almacenamiento por clúster (cuando los datos se distribuyen uniformemente por la clave de partición)	4 PiB
Conexiones al clúster	El valor inferior de 300 000 o el número de particiones x la cuota de conexión asociada a la vCPU por partición
UnSharded tamaño de la colección	32 TiB
Tamaño de la colección con particiones (cuando los datos se distribuyen uniformem ente mediante la clave de partición)	1 PB
Bases de datos por clúster	10 000
UnSharded colecciones por clúster	100 000
Colecciones por clúster con particiones	1 000
Usuarios por clúster	100
Escritura en un único comando por lotes	100 000

Recurso	Cuota
Índices por colección	64
Profundidad de anidamiento de documentos	100 niveles
Tamaño del documento	16 MB
Tamaño de clave de índice	2048 bytes
Claves de un índice compuesto	32

Cuotas de particiones de clústeres elásticos

En la siguiente tabla se describen las cuotas máximas de fragmentos para los clústeres elásticos de Amazon DocumentDB.

Recurso	Cuota
vCPU por instancia de partición	64
Instancias por partición	16
Almacenamiento por partición	128 TiB
Almacenamiento por colección y por partición	32 TiB

Cuotas de CPU, memoria, conexión y cursor de los clústeres elásticos por fragmento

En la siguiente tabla se describen los límites máximos de CPU, memoria, conexión y cursor en las particiones de los clústeres elásticos de Amazon DocumentDB.

VCPUs por partición	Memoria de la instancia (GiB)	Cuota de conexión	Cuota de cursores
2	16	1700	450

VCPUs por partición	Memoria de la instancia (GiB)	Cuota de conexión	Cuota de cursores
4	32	3500	450
8	64	7100	450
16	128	14200	760
32	256	28400	1520
48	384	30000	2280
64	512	30000	3040

Solución de problemas de Amazon DocumentDB

En las siguientes secciones, se proporciona información acerca de cómo solucionar los problemas que puedan surgir al utilizar Amazon DocumentDB (compatible con MongoDB).

Temas

- Solución de problemas de conectividad
- Índices de solución de problemas
- Solución de problemas de rendimiento y utilización de recursos
- Recopilación de elementos no utilizados en Amazon DocumentDB
- Códigos de error

Solución de problemas de conectividad

¿Tiene problemas para conectarse? A continuación se muestran algunos escenarios comunes y cómo resolverlos.

Temas

- No se puede conectar a un punto de conexión de Amazon DocumentDB
- Comprobación de una conexión a una instancia de Amazon DocumentDB
- Conexión a un punto de conexión no válido
- La configuración del controlador afecta al número de conexiones

No se puede conectar a un punto de conexión de Amazon DocumentDB

Al intentar conectarse a Amazon DocumentDB, el siguiente es uno de los mensajes de error más comunes que podría encontrarse.

```
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
```

Este mensaje de error normalmente significa que el cliente (el intérprete de comandos de mongo, en este ejemplo) no puede obtener acceso al punto de conexión de Amazon DocumentDB. Esto puede deberse a varios motivos:

Temas

- Conexión desde puntos de conexión públicos
- Cross-Region conexiones
- Conexión desde diferentes Amazon VPC
- Conexiones entrantes en bloques del grupo de seguridad
- Problema de preferencia de lectura del controlador Java Mongo

Conexión desde puntos de conexión públicos

Está intentando conectarse a un clúster de Amazon DocumentDB directamente desde un portátil o un equipo de desarrollo local.

Se producirá un error al intentar conectarse a un clúster de Amazon DocumentDB directamente desde un dispositivo de punto de conexión público, como una laptop o una máquina de desarrollo local. Amazon DocumentDB solo está diseñado para una nube privada virtual (VPC) y actualmente no admite puntos de conexión públicos. Por lo tanto, no puede conectarse directamente a su clúster de Amazon DocumentDB desde un portátil o un entorno de desarrollo local que esté fuera de la VPC.

Para conectarse a un clúster de Amazon DocumentDB desde fuera de una VPC de Amazon, puede utilizar un túnel de SSH. Para obtener más información, consulte Conexión a un clúster de Amazon DocumentDB desde fuera de una Amazon VPC. Además, si su entorno de desarrollo se encuentra en una Amazon VPC diferente, también puede usar la interconexión de VPC y conectarse a su clúster de Amazon DocumentDB desde otra Amazon VPC de la misma región o de una región diferente.

Cross-Region conexiones

Está intentando conectarse a un clúster de Amazon DocumentDB en otra región.

Si intenta conectarse a un clúster de Amazon DocumentDB desde una instancia de Amazon EC2 en una región distinta a la región del clúster (por ejemplo, si intenta conectarse a un clúster en región Este de EE. UU. (Norte de Virginia) (us-east-1) desde la región Oeste de EE. UU. (Oregón) (us-west-2), la conexión fallará.

Para verificar la región del clúster de Amazon DocumentDB, ejecute el siguiente comando. La región está en el punto de conexión.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].Endpoint'
```

La salida de esta operación será similar a lo que se indica a continuación.

```
[
    "sample-cluster.node.us-east-1.docdb.amazonaws.com"
]
```

Para verificar la región de la instancia EC2, ejecute el siguiente comando.

```
 aws ec2 describe-instances \
     --query 'Reservations[*].Instances[*].Placement.AvailabilityZone'
```

La salida de esta operación será similar a lo que se indica a continuación.

```
[
    [
        "us-east-1a"
    ]
]
```

Conexión desde diferentes Amazon VPC

Está intentando conectarse a un clúster de Amazon DocumentDB desde una VPC distinta de la VPC de Amazon en la que se ha implementado el clúster.

Si tanto el clúster de Amazon DocumentDB como la instancia de Amazon EC2 están en la misma Amazon VPC Región de AWS, pero no en la misma Amazon, no podrá conectarse directamente a su clúster de Amazon DocumentDB a menos que esté habilitada la interconexión de VPC entre las dos VPC de Amazon.

Para comprobar la Amazon VPC de su instancia de Amazon DocumentDB, ejecute el siguiente comando.

```
aws docdb describe-db-instances \
   --db-instance-identifier sample-instance \
   --query 'DBInstances[*].DBSubnetGroup.VpcId'
```

Para comprobar la Amazon VPC de su instancia de Amazon EC2, ejecute el siguiente comando.

```
aws ec2 describe-instances \
   --query 'Reservations[*].Instances[*].VpcId'
```

Conexiones entrantes en bloques del grupo de seguridad

Está intentando conectarse a un clúster de Amazon DocumentDB, pero el grupo de seguridad del clúster no admite conexiones entrantes en el puerto del clúster (el puerto predeterminado es el 27017).

Supongamos que el clúster de Amazon DocumentDB y la instancia Amazon EC2 se encuentran en la misma región y la VPC de Amazon, y utiliza el mismo grupo de seguridad de Amazon VPC. Si no puede conectarse al clúster de Amazon DocumentDB, es probable que la causa sea que el grupo de seguridad (es decir, el firewall) del clúster no permite las conexiones entrantes en el puerto que eligió para el clúster de Amazon DocumentDB (el puerto predeterminado es el 27017).

Para verificar el puerto del clúster de Amazon DocumentDB, ejecute el siguiente comando.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[DBClusterIdentifier,Port]'
```

Para obtener el grupo de seguridad de Amazon DocumentDB de su clúster, ejecute el siguiente comando.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[VpcSecurityGroups[*],VpcSecurityGroupId]'
```

Para comprobar las reglas de entrada del grupo de seguridad, consulte los temas siguientes de la documentación de Amazon EC2:

- Autorización del tráfico de entrada para sus instancias de Linux
- Autorización del tráfico de entrada para sus instancias de Windows

Problema de preferencia de lectura del controlador Java Mongo

No se respetan las preferencias de lectura del cliente y algunos clientes no pueden escribir en Amazon DocumentDB después de una conmutación por error a menos que reinicien.

Este problema, descubierto por primera vez en Java Mongo Driver 3.7.x, se produce cuando un cliente establece una conexión con Amazon DocumentDB mediante `MongoClientSettings` y, específicamente, al encadenar el método `applyToClusterSettings`. La configuración del MongoClient clúster se puede definir mediante diferentes métodos, como, y. `hosts()` `requiredReplicaSetName()` `mode()`

Cuando el cliente especifica solo un host en el método `hosts()`, el modo se establece en `ClusterConnectionMode.SINGLE` lugar de `ClusterConnectionMode.MULTIPLE`. Esto hace que el cliente ignore la preferencia de lectura y solo se conecte al servidor configurado en `hosts()`. Por lo tanto, incluso si la configuración del cliente se inicializa como se muestra a continuación, todas las lecturas seguirán yendo a la principal en lugar de a la secundaria.

```
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 de conmutación por error

Con la configuración de conexión del cliente anterior, si se produce una conmutación por error y se retrasa la actualización del registro de DNS en el punto de conexión del escritor del clúster, el cliente

seguirá intentando realizar escrituras en el antiguo escritor (que ahora se lee tras la conmutación por error). Esto provoca un error en el servidor (no en el principal) que el controlador de Java no gestiona adecuadamente (el problema aún se está investigando). Por lo tanto, el cliente puede quedar en mal estado hasta que se reinicie el servidor de aplicaciones, por ejemplo.

Existen dos soluciones alternativas para ello:

- Los clientes que se conecten a Amazon DocumentDB mediante una cadena de conexión no tendrán este problema, ya que `ClusterConnectionMode` se establecerá en `MULTIPLE` al configurar la preferencia de lectura.

  ```
  MongoClientURI mongoClientURI = new MongoClientURI("mongodb://usr:pass:cluster-
  endpoint:27317/test?ssl=false&replicaSet=rs0&readpreference=secondaryPreferred");
  MongoClient mongoClient = MongoClients.create(mongoClientURI.getURI());
  ```

  O usar el constructor `MongoClientSettings` con el método `applyConnectionString`.

  ```
  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);
  ```

- Establecido `ClusterConnectionMode` explícitamente en `MULTIPLE`. Esto solo es necesario cuando se usa `applyToClusterSettings` y `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);
```

Comprobación de una conexión a una instancia de Amazon DocumentDB

Puede comprobar la conexión a un clúster mediante las herramientas habituales de Windows o de Linux.

En un terminal de Linux o Unix, puede comprobar la conexión escribiendo lo siguiente (sustituya `cluster-endpoint` por el punto de conexión y `port` por el puerto de la instancia):

```
nc -zv cluster-endpoint port
```

A continuación se muestra un ejemplo de una operación y el valor devuelto:

```
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!
```

Conexión a un punto de conexión no válido

Si se conecta a un clúster de Amazon DocumentDB y utiliza un punto de conexión del clúster que no es válido, aparece un error similar al siguiente.

```
mongo --ssl \
   --host sample-cluster.node.us-east-1.docdb.amazonaws.com:27017 \
   --sslCAFile global-bundle.pem \
   --username <user-name> \
   --password <password>
```

El resultado tendrá este aspecto:

```
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
```

Para obtener el punto de conexión válido de un clúster, utilice el comando siguiente:

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[Endpoint,Port]'
```

Para obtener el punto de conexión válido de una instancia, utilice el comando siguiente:

```
aws docdb describe-db-instances \
   --db-instance-identifier sample-instance \
   --query 'DBInstances[*].[Endpoint.Address,Endpoint.Port]'
```

Para obtener más información, consulte Descripción de los puntos de conexión de Amazon DocumentDB.

La configuración del controlador afecta al número de conexiones

Al utilizar el controlador de cliente para conectarse a un clúster de Amazon DocumentDB, es importante tener en cuenta el parámetro de configuración `maxPoolSize`. La configuración `maxPoolSize` determina el número máximo de conexiones que el controlador de cliente mantendrá en su grupo de conexiones.

Índices de solución de problemas

En los temas siguientes, se explica lo que debe hacer si se produce un error durante la creación de un índice o un índice en segundo plano.

Temas

- Error al crear un índice
- El índice en segundo plano genera problemas de latencia y falla
- Sobrecarga del índice de base de datos

Error al crear un índice

Amazon DocumentDB utiliza el almacenamiento local de una instancia durante el proceso de creación de índices. Puede supervisar el uso de este disco mediante la FreeLocalStorage CloudWatch métrica (`CloudWatch -> Metrics -> DocDB -> Instance Metrics`). Cuando la generación de un índice consume todo el espacio del disco local y no finaliza, se muestra un error. Cuando migre datos a Amazon DocumentDB, conviene que cree los índices primero y, a continuación, inserte los datos. Para obtener más información sobre las estrategias de migración y la creación de índices, consulte Migración y actualización de Amazon DocumentDB en la documentación de Amazon DocumentDB y la entrada de blog: Migrar de MongoDB a Amazon DocumentDB utilizando el método sin conexión.

Al crear índices en un clúster existente, si la creación del índice tarda más de lo esperado o no funciona, amplíe la instancia para crear el índice y, a continuación, reduzca la escala a la baja una vez creado el índice. Amazon DocumentDB le permite escalar rápidamente el tamaño de las instancias en cuestión de minutos mediante el Consola de administración de AWS o el. AWS CLI Para obtener más información, consulte Administración de clases de instancias. Dado que los precios de las instancias se calculan por segundos, solo pagará por los segundos en que esté utilizando los recursos.

El índice en segundo plano genera problemas de latencia y falla

Las compilaciones de índices en segundo plano en Amazon DocumentDB no se inician hasta que todas las consultas de la instancia principal que se iniciaron antes de que se iniciara la creación del índice terminen de ejecutarse. Si hay una consulta de larga duración, las compilaciones de índices en segundo plano se bloquearán hasta que finalice la consulta y, por lo tanto, pueden tardar más de lo esperado en completarse. Esto es válido incluso si las colecciones están vacías.

Las compilaciones de índices en primer plano no muestran el mismo comportamiento de bloqueo. En cambio, las compilaciones de índices en primer plano controlan exclusivamente la colección hasta que se complete la creación del índice. Para crear índices en una colección vacía y evitar bloquear las consultas que se ejecutan durante mucho tiempo, utilice compilaciones de índices en primer plano.

> **Note**
>
> Amazon DocumentDB solo permite una operación de creación de índice en segundo plano en una colección al mismo tiempo. Si se producen operaciones de DDL (Lenguaje de

definición de datos) como `createIndex()` o `dropIndex()` en la misma colección durante la operación de creación de un índice en segundo plano, esta operación producirá un error.

Sobrecarga del índice de base de datos

Amazon DocumentDB usa el control de Multi-Version concurrencia (MVCC) para administrar las transacciones simultáneas. Cuando los documentos se eliminan o actualizan, sus versiones anteriores permanecen en las colecciones e índices como versiones «inactivas». El proceso de recopilación de elementos no utilizados recupera automáticamente el espacio de estas versiones inactivas para futuras operaciones.

La sobrecarga de índices se produce cuando el tamaño de los índices de una colección aumenta debido a la acumulación de índices inactivos u obsoletos o a la fragmentación de las páginas. El porcentaje informado representa la cantidad de espacio de índice que pueden utilizar las entradas de índice futuras. Esta sobrecarga consume espacio tanto en la memoria caché como en el almacenamiento del búfer. Si desea eliminar la sobrecarga, tendrá que recopilar los índices.

Example Ejemplo

Ejecute el siguiente comando para determinar el almacenamiento no utilizado del índice:

```
db.coll.aggregate({$indexStats:{}});
```

El resultado tendrá un aspecto parecido al siguiente:

```
{
    "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
    }
}
```

Puede reconstruir los índices sin tiempo de inactividad mediante el comando `reIndex`, que requiere escanear toda la colección. Consulte Mantenimiento de índices mediante ReIndex .

Solución de problemas de rendimiento y utilización de recursos

En esta sección se proporcionan preguntas y soluciones para los problemas de diagnóstico frecuentes en las implementaciones de Amazon DocumentDB. Los ejemplos que se proporcionan utilizan el intérprete de comandos de mongo y se limitan a una instancia individual. Para encontrar un punto de conexión de instancia, consulte Descripción de los puntos de conexión de Amazon DocumentDB.

Temas

- ¿Cómo determino el número de operaciones de inserción, actualización y eliminación realizadas en mi colección a través de la API de Mongo?
- ¿Cómo analizo el rendimiento de la memoria caché?
- ¿Cómo puedo encontrar y terminar las consultas que tardan mucho en ejecutarse o se bloquean?
- ¿Cómo puedo ver un plan de consulta y optimizar una consulta?
- ¿Cómo puedo ver un plan de consultas en clústeres elásticos?
- ¿Cómo puedo ver todas las operaciones en ejecución en una instancia?
- ¿Cómo sé cuándo una consulta está avanzando?
- ¿Cómo determino por qué de repente un sistema se ejecuta lentamente?
- ¿Cómo determino la causa del uso elevado de la CPU en una o varias instancias de clúster?
- ¿Cómo determino los cursores abiertos en una instancia?
- ¿Cómo determino la versión actual del motor de Amazon DocumentDB?
- ¿Cómo analizo el uso de los índices e identifico los índices no utilizados?

- ¿Cómo identifico los índices que faltan?
- ¿Cómo determino la sobrecarga de colecciones de bases de datos?
- Resumen de consultas útiles

¿Cómo determino el número de operaciones de inserción, actualización y eliminación realizadas en mi colección a través de la API de Mongo?

Para ver el número de operaciones de inserción, actualización y eliminación realizadas en una colección determinada, ejecute el siguiente comando en esa colección:

```
db.collection.stats()
```

La salida de este comando describe lo siguiente en su campo `opCounters`:

- num DocsIns: el número de documentos insertados en esta colección. Incluye los documentos insertados mediante los comandos `insert` y `insertMany`, así como los documentos insertados mediante actualizar o insertar.
- num DocsUpd: el número de actualizaciones de documentos de esta colección. Incluye los documentos actualizados mediante los comandos `update` y `findAndModify`.
- num DocsDel: el número de documentos eliminados de esta colección. Incluye los documentos eliminados mediante los comandos `deleteOne`, `deleteMany`, `remove` y `findAndModify`.
- lastReset: hora a la que se restablecieron estos contadores por última vez. Las estadísticas que proporciona este comando se restablecen starting/stopping al agrupar o escalar up/down la instancia.

A continuación, se muestra una salida de ejemplo de ejecución de `db.collection.stats()`.

```
{
    "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)
}
```

Este comando de estadística debe usarse al ver los contadores específicos de la colección para las operaciones de inserción, actualización y eliminación a través de la API de Mongo. Otra forma de ver los contadores de operaciones específicas de una colección es habilitar la auditoría de DML. Se puede ver en Supervisión de Amazon DocumentDB con CloudWatch el número de operaciones de inserción, actualización y eliminación en todas las colecciones durante intervalos de un minuto.

¿Cómo analizo el rendimiento de la memoria caché?

El análisis del rendimiento de la memoria caché puede proporcionar información sobre la eficiencia de la recuperación de datos y el rendimiento del sistema, y se basa en la cantidad de datos que se leen del disco en comparación con la memoria caché. Proporcionamos estadísticas de la memoria caché sobre el número de visitas a la memoria caché (datos leídos de la memoria caché) y pérdidas de memoria caché (datos que no se encuentran en la memoria caché y se leen en el disco) para obtener información sobre el rendimiento de la memoria caché. Las estadísticas de la memoria caché de una colección específica se pueden encontrar ejecutando el siguiente comando en esa colección:

```
db.collection.stats()
```

Los valores del campo `cacheStats` del resultado de este comando proporcionan las estadísticas de la memoria caché de la colección, así como las estadísticas de la memoria caché totales de los índices creados en la colección. Estas estadísticas se muestran a continuación:

- **`collBlksHit`**: el número de bloques leídos de la memoria caché durante las operaciones de esta colección.
- **`collBlksRead`**: el número de bloques leídos del disco (pérdidas de memoria caché) durante las operaciones de esta colección.
- **`collHitRatio`**: la proporción de aciertos de la memoria caché de esta colección (`100 * [collBlksHit / (collBlksHit + collBlksRead)]`).
- **`idxBlksHit`**: el número de bloques leídos de la memoria caché para cualquier índice creado en esta colección.
- **`idxBlksRead`**: el número de bloques leídos del disco (pérdidas de memoria caché) para cualquier índice creado en esta colección.
- **`idxHitRatio`**: la proporción de aciertos de la memoria caché creados en esta colección (`100 * [idxBlksHit / (idxBlksHit + idxBlksRead)]`).
- **`lastReset`**: hora a la que se restablecieron estas estadísticas por última vez. Las estadísticas proporcionadas por `db.collection.stats()` se restablecen cuando starting/stopping se agrupa o escala up/down la instancia.

También se puede encontrar un desglose de los campos `idxBlksHit` y `idxBlksRead` de cada índice mediante el comando `indexStats`. Puede encontrar las estadísticas de memoria caché específicas del índice ejecutando el siguiente comando:

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

Para cada índice, se encuentran las siguientes estadísticas de memoria caché en el campo `cacheStats`:

- **`blksHit`**: el número de bloques leídos de la memoria caché para este índice.
- **`blksRead`**: el número de bloques leídos del disco para este índice.
- **`blksHitRatio`**: la proporción de aciertos de la memoria caché redondeada a cuatro decimales, calculada mediante `100 * [blksHit / (blksHit + blksRead)]`.

¿Cómo puedo encontrar y terminar las consultas que tardan mucho en ejecutarse o se bloquean?

Es posible que las consultas del usuario se ejecuten lentamente debido a que el plan de consultas es inadecuado o a que se bloquean debido a la contención de recursos.

Para buscar consultas de ejecución prolongada que se ralenticen a causa de un plan de consultas inadecuado o consultas que se bloquean debido a la contención de recursos, utilice el comando `currentOp`. Puede filtrar el comando para acotar la lista de las consultas relevantes que deben terminarse. Debe tener asociado `opid` a la consulta de ejecución prolongada para poder terminarla.

En la siguiente consulta, se utiliza el comando `currentOp` para ver todas las consultas que están bloqueadas o que se llevan ejecutando durante más de 10 segundos.

```
db.adminCommand({
    aggregate: 1,
    pipeline: [
        {$currentOp: {}},
        {$match:
            {$or: [
                {secs_running: {$gt: 10}},
                {WaitState: {$exists: true}}]}},
        {$project: {_id:0, opid: 1, secs_running: 1}}],
    cursor: {}
});
```

A continuación, puede limitar la consulta para encontrar el `opid` de una consulta que lleva ejecutándose durante más de 10 segundos y terminarla.

Para encontrar y terminar una consulta que lleva ejecutándose durante más de 10 segundos

1. Busque el `opid` de la consulta.

   ```
   db.adminCommand({
       aggregate: 1,
       pipeline: [
           {$currentOp: {}},
           {$match:
               {$or:
                   [{secs_running: {$gt: 10}},
                    {WaitState: {$exists: true}}]}}],
   ```

```
    cursor: {}
});
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
                "opid" : 24646,
                "secs_running" : 12
            }
        ],
        "id" : NumberLong(0),
        "ns" : "admin.$cmd"
    },
    "ok" : 1
}
```

2. Termine la consulta con la operación `killOp`.

```
db.adminCommand({killOp: 1, op: 24646});
```

¿Cómo puedo ver un plan de consulta y optimizar una consulta?

Si una consulta se ejecuta lentamente, podría deberse a que la ejecución de la consulta requiere un examen completo de la colección para elegir los documentos pertinentes. A veces, la creación de los índices adecuados permite que la consulta se ejecute con mayor rapidez. Para detectar este escenario y decidir en qué campos se deben crear los índices, utilice el comando `explain`.

> **Note**
>
> Amazon DocumentDB emula la API MongoDB 3.6 en un motor de base de datos personalizada específicamente que utiliza un sistema de almacenamiento distribuido, tolerante a fallos y de recuperación automática. Como resultado, los planes de consulta y la salida de `explain()` pueden diferir entre Amazon DocumentDB y MongoDB. Los clientes que deseen controlar su plan de consulta pueden utilizar el operador `$hint` para aplicar la selección de un índice preferido.

Ejecute la consulta que desee mejorarse en el comando `explain` del modo siguiente.

```
db.runCommand({explain: {<query document>}})
```

A continuación se muestra un ejemplo de operación.

```
db.runCommand({explain:{
    aggregate: "sample-document",
    pipeline: [{$match: {x: {$eq: 1}}}],
    cursor: {batchSize: 1}}
});
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "db.test",
        "winningPlan" : {
            "stage" : "COLLSCAN"
        }
    },
    "serverInfo" : {
        "host" : "...",
        "port" : ...,
        "version" : "..."
    },
    "ok" : 1
}
```

El resultado anterior indica que la etapa `$match` requiere examinar todas las colecciones y comprobar si el campo `"x"` de cada documento es igual a 1. Si hay muchos documentos en la colección, el examen de la consulta (y, por tanto, el rendimiento general de la consulta) es muy lento. Por consiguiente, la presencia de `"COLLSCAN"` en el resultado del comando `explain` indica que el rendimiento de la consulta se puede mejorar creando los índices adecuados.

En este ejemplo, la consulta comprueba si el campo `"x"` es igual a 1 en todos los documentos. Por tanto, la creación de un índice en el campo `"x"` permite que la consulta evite el examen completo de la colección y utilice el índice para devolver los documentos pertinentes antes.

Después de crear un índice en el campo `"x"`, el resultado de `explain` es el siguiente.

```
{
    "queryPlanner" : {
         "plannerVersion" : 1,
         "namespace" : "db.test",
         "winningPlan" : {
             "stage" : "IXSCAN",
             "indexName" : "x_1",
             "direction" : "forward"
         }
    },
    "serverInfo" : {
        "host" : "...",
        "port" : ...,
        "version" : "..."
    },
    "ok" : 1
}
```

La creación de un índice en el campo `"x"` permite que la etapa `$match` utilice un análisis de índices para reducir el número de documentos en los que debe evaluarse el predicado `"x = 1"`.

En colecciones pequeñas, el procesador de consultas de Amazon DocumentDB puede optar por no utilizar un índice si las ventajas para el rendimiento son insignificantes.

¿Cómo puedo ver un plan de consultas en clústeres elásticos?

Para examinar un plan de consultas en clústeres elásticos, utilice el comando `explain`. A continuación, se muestra un ejemplo de operación `explain` en una consulta de búsqueda dirigida a una colección con partición:

```
db.runCommand(
   {
     explain: { find: "cities", filter: {"name": "Seoul"}}
   }
)
```

> **Note**
>
> Amazon DocumentDB emula MongoDB en un motor de base de datos personalizada. Como resultado, los planes de consulta y la salida de `explain()` pueden diferir entre Amazon

DocumentDB y MongoDB. Los clientes que deseen controlar su plan de consulta pueden utilizar el operador `$hint` para aplicar la selección de un índice preferido.

La salida de esta operación será similar a lo que se indica a continuación (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)
}
```

El resultado anterior muestra el plan de consulta para la consulta `find` en un clúster de tres particiones. Cada partición tiene varias particiones de datos que pueden tener diferentes etapas de entrada. En este ejemplo, se ejecuta un “COLLSCAN” (un escaneo de colecciones) en todas las particiones antes de fusionar los resultados en la etapa “PARTITION_MERGE” de cada partición. A continuación, los resultados de las particiones se combinan en la etapa “SHARD_MERGE” antes de enviarlos de vuelta al cliente.

¿Cómo puedo ver todas las operaciones en ejecución en una instancia?

Como un usuario o usuario principal, a menudo desea ver una lista de todas las operaciones que se están ejecutando actualmente en una instancia con fines de diagnóstico y para solucionar problemas. (Para obtener información acerca de cómo administrar los usuarios, consulte Administración de usuarios de Amazon DocumentDB).

Con el intérprete de comandos de `mongo`, puede utilizar la siguiente consulta para ver una lista de todas las operaciones en ejecución en una instancia de Amazon DocumentDB.

```
db.adminCommand({currentOp: 1, $all: 1});
```

La consulta devuelve la lista completa de todas las consultas del usuario y las tareas internas del sistema que se están realizando actualmente en la instancia.

La salida de esta operación será similar a lo que se indica a continuación (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
}
```

Los siguientes valores son válidos para el campo `"desc"`:

- **`INTERNAL`**: tareas internas del sistema, como tareas de limpieza del cursor o limpieza de usuarios obsoletos.
- **`TTLMonitor`**: el hilo de supervisión de tiempo de vida (TTL). Su estado de funcionamiento se refleja en el campo `"active"`.
- **`GARBAGE_COLLECTION`**: el subproceso del recolector de elementos no utilizados interno.
- **`CONN`**: la consulta del usuario.

- **CURSOR**: la operación consiste en un cursor inactivo que espera a que el usuario llame al comando "GetMore" para obtener el siguiente lote de resultados. En este estado, el cursor consume memoria, pero no consume capacidad de cálculo.

El resultado anterior también muestra todas las consultas del usuario que se ejecutan en el sistema. Cada consulta del usuario se ejecuta en el contexto de una base de datos y una colección. La unión de las dos se denomina espacio de nombres. El espacio de nombres de cada consulta del usuario está disponible en el campo `"ns"`.

A veces, es necesario ver una lista de todas las consultas del usuario que se están ejecutando en un determinado espacio de nombres. Por tanto, el resultado anterior debe filtrarse en el campo `"ns"`. A continuación, se muestra una consulta de ejemplo para conseguir el resultado que se va a filtrar. La consulta muestra todas las consultas del usuario que se están ejecutando actualmente en la base de datos `"db"` y la colección `"test"` (es decir, el espacio de nombres `"db.test"`).

```
db.adminCommand({aggregate: 1,
    pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
               {$match: {ns: {$eq: "db.test"}}}],
    cursor: {}
});
```

Como usuario principal del sistema, puede ver las consultas de todos los usuarios y también todas las tareas internas del sistema. Los demás usuarios solo pueden ver sus consultas respectivas.

Si el número total de consultas y tareas internas del sistema supera el tamaño del cursor batch predeterminado, el intérprete de comandos de `mongo` genera automáticamente un objeto de iteración `'it'` para ver el resto de los resultados. Siga ejecutando el comando `'it'` hasta que se hayan agotado todos los resultados.

¿Cómo sé cuándo una consulta está avanzando?

Es posible que las consultas del usuario se ejecuten lentamente debido a que el plan de consultas es inadecuado o a que se bloquean debido a la contención de recursos. La depuración de estas consultas es un proceso de varios pasos en el que puede ser necesario realizar el mismo paso varias veces.

El primer paso de la depuración es mostrar todas las consultas que tardan mucho tiempo en ejecutarse o que se bloquean. La siguiente consulta muestra todas las consultas del usuario que se han ejecutado durante más de 10 segundos o que están esperando recursos.

```
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: {}
                });
```

Repita la consulta anterior de forma periódica para determinar si la lista de consultas cambia e identificar la consulta de ejecución prolongada o las consultas bloqueadas.

Si el documento de salida de la consulta en cuestión tiene un campo `WaitState`, indica que la contención de recursos es la razón por la que la consulta tarda mucho en ejecutarse o se bloquea. La contención de recursos puede deberse a I/O tareas internas del sistema u otras consultas de los usuarios.

La salida de esta operación será similar a lo que se indica a continuación (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 puede ser un cuello de botella si muchas consultas de diferentes colecciones se ejecutan simultáneamente en la misma instancia o si el tamaño de la instancia es demasiado pequeño para el conjunto de datos en el que se ejecuta la consulta. Si las consultas son consultas de solo lectura, puede mitigar la situación anterior separando las consultas de cada colección en réplicas diferentes. En las actualizaciones simultáneas en diferentes colecciones o cuando el tamaño de la instancia es demasiado pequeño para el conjunto de datos, puede mitigarla ampliando la instancia.

Si la contención de recursos se debe a otras consultas del usuario, el campo `"blockedOn"` del documento de salida contendrá el `"opid"` de la consulta que afecta a esta consulta. Mediante el `"opid"`, siga la cadena de campos `"WaitState"` y `"blockedOn"` de todas las consultas para encontrar la consulta de la parte superior de la cadena.

Si la tarea de la parte superior de la cadena es una tarea interna, la única forma de mitigar este problema sería finalizar la consulta y volver a ejecutarla más adelante.

A continuación, se muestra un resultado de ejemplo en el que la consulta de búsqueda está bloqueada en un bloqueo de la colección propiedad de otra tarea.

```
{
    "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
}
```

Si "WaitState" tiene los valores "Latch", "SystemLock", "BufferLock", "BackgroundActivity" o "Other", la contención de recursos se debe a tareas internas del sistema. Si la situación continúa durante mucho tiempo, la única forma de mitigar este problema sería finalizar la consulta y volver a ejecutarla más adelante.

¿Cómo determino por qué de repente un sistema se ejecuta lentamente?

A continuación, se indican algunos motivos frecuentes de la ralentización del sistema:

- Contención excesiva de recursos entre consultas simultáneas
- El número de consultas simultáneas activas aumenta con el tiempo
- Tareas internas del sistema como "GARBAGE_COLLECTION"

Para monitorizar el uso del sistema a lo largo del tiempo, ejecute la siguiente consulta "currentOp" periódicamente y envíe los resultados a un almacén externo. La consulta cuenta la cantidad de

consultas y operaciones en cada espacio de nombres del sistema. A continuación, puede analizar los resultados de uso del sistema para conocer la carga del sistema y tomar las medidas adecuadas.

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
                            {$group: {_id: {desc: "$desc", ns: "$ns", WaitState:
 "$WaitState"}, count: {$sum: 1}}}],
                 cursor: {}
                });
```

Esta consulta devuelve la suma de todas las consultas que se ejecutan en cada espacio de nombres, todas las tareas internas del sistema y el número único de estados de espera (si hay alguno) por espacio de nombres.

La salida de esta operación será similar a lo que se indica a continuación (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
}
```

En el resultado anterior, hay dos consultas de usuario en el espacio de nombres `"db.test"` que están bloqueadas en el bloqueo de colección: una consulta en el espacio de nombres `"admin.$cmd"` y una tarea interna `"TTLMonitor"`.

Si el resultado indica muchas consultas con estados de espera de bloqueo, consulte ¿Cómo puedo encontrar y terminar las consultas que tardan mucho en ejecutarse o se bloquean?

¿Cómo determino la causa del uso elevado de la CPU en una o varias instancias de clúster?

Las secciones siguientes pueden ayudarle a identificar la causa del uso elevado de la CPU en una instancia. Los resultados pueden variar en función de la carga de trabajo.

- Para determinar por qué una instancia de repente se ejecuta lentamente, consulte ¿Cómo determino por qué de repente un sistema se ejecuta lentamente?
- Para identificar y terminar consultas de ejecución largas en una instancia determinada, consulte ¿Cómo puedo encontrar y terminar las consultas que tardan mucho en ejecutarse o se bloquean?
- Para saber si una consulta avanza, consulte ¿Cómo sé cuándo una consulta está avanzando?
- Para determinar por qué una consulta tarda mucho tiempo en ejecutarse, consulte ¿Cómo puedo ver un plan de consulta y optimizar una consulta?
- Para realizar un seguimiento de las consultas de ejecución prolongada con el tiempo, consulte Creación de perfiles de operaciones en Amazon DocumentDB.

En función del motivo del uso elevado de la CPU en la instancia, puede resultar útil realizar una o varias de las siguientes acciones.

- Si la instancia principal presenta un uso elevado de la CPU, pero las instancias de réplica no, considere la posibilidad de distribuir el tráfico de lectura entre las réplicas a través de la configuración de preferencia de lectura del cliente (por ejemplo, `secondaryPreferred`). Para obtener más información, consulte Conexión a Amazon DocumentDB como conjunto de réplicas.

Si se utilizan réplicas para las lecturas, se puede aprovechar mejor los recursos del clúster permitiendo que la instancia principal procese más tráfico de escritura. Las lecturas de las réplicas presentan consistencia final.

- Si el uso elevado de la CPU se debe a la carga de trabajo de escritura, cambiar el tamaño de las instancias del clúster a un tipo de instancia mayor aumenta el número de núcleos de la CPU disponibles para atender la carga de trabajo. Para obtener más información, consulte Administración de instancias de Amazon DocumentDB y Especificaciones de clases de instancias.
- Si todas las instancias del clúster presentan un uso elevado de la CPU y la carga de trabajo utiliza réplicas para las lecturas, añadir más réplicas al clúster aumenta los recursos disponibles para el tráfico de lectura. Para obtener más información, consulte Agregación de una instancia de Amazon DocumentDB a un clúster.

¿Cómo determino los cursores abiertos en una instancia?

Cuando está conectado a una instancia Amazon DocumentDB, puede utilizar el comando `db.runCommand("listCursors")` para enumerar los cursores abiertos en esa instancia. Hay una cuota de hasta 4560 cursores activos abiertos en un momento dado en una instancia determinada de Amazon DocumentDB, según el tipo de instancia. Cierre los cursores que ya no estén en uso porque los cursores utilizan los recursos de una instancia. Para obtener más información, consulte Cuotas de Amazon DocumentDB.

```
db.runCommand("listCursors")
```

¿Cómo determino la versión actual del motor de Amazon DocumentDB?

Para determinar la versión actual del motor de Amazon DocumentDB, ejecute el siguiente comando.

```
db.runCommand({getEngineVersion: 1})
```

La salida de esta operación será similar a lo que se indica a continuación (formato JSON).

```
{ "engineVersion" : "2.x.x", "ok" : 1 }
```

> **Note**
>
> La versión del motor de Amazon DocumentDB 3.6 es 1.x.x, la versión del motor de Amazon DocumentDB 4.0 es 2.x.x, la versión del motor de Amazon DocumentDB 5.0 es 3.x.x y la versión del motor de Amazon DocumentDB 8.0 es 4.x.x.

¿Cómo analizo el uso de los índices e identifico los índices no utilizados?

Para identificar los índices de una colección determinada, ejecute el siguiente comando:

```
db.collection.getIndexes()
```

Para analizar cuántos índices se utilizan durante las operaciones realizadas en las colecciones, se pueden utilizar los comandos `collStats` y `indexStats`. Para ver el número total de escaneos realizados con índices (escaneos de índices) en comparación con el número de escaneos realizados sin un índice (escaneos de colecciones), ejecute el siguiente comando:

```
db.collection.stats()
```

El resultado de este comando incluye los valores que se muestran a continuación:

- **`idxScans`**: el número de escaneos realizados en esta colección mediante un índice.
- **`collScans`**: el número de escaneos realizados en esta colección sin un índice. Estos escaneos habrían implicado revisar los documentos de la colección uno por uno.
- **`lastReset`**: hora a la que se restablecieron estos contadores por última vez. Las estadísticas que proporciona este comando se restablecen al agrupar o escalar la instancia. starting/stopping up/down

En el resultado del siguiente comando se muestra un desglose del uso de cada índice. Se recomienda identificar y eliminar con regularidad los índices no utilizados para mejorar el rendimiento y reducir los costos, ya que así se eliminan los gastos de procesamiento, almacenamiento y I/Os mantenimiento innecesarios.

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

El resultado de este comando proporciona los siguientes valores para cada índice creado en la colección:

- **`ops`**: el número de operaciones que utilizaron el índice. Si la carga de trabajo se ha ejecutado durante un tiempo suficientemente largo y está seguro de que se encuentra en un estado estable, un valor `ops` de cero indicaría que el índice no se utiliza en absoluto.
- **`numDocsRead`**: el número de documentos leídos durante las operaciones que utilizan este índice.
- **`since`**: el tiempo transcurrido desde que Amazon DocumentDB comenzó a recopilar estadísticas sobre el uso del índice, que suele ser el valor transcurrido desde la última acción de reinicio o mantenimiento de la base de datos.
- **`size`**: tamaño de este archivo en bytes.

El siguiente ejemplo es una muestra del resultado de ejecutar el comando anterior:

```
{
    "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" : ...
    }
}
```

Para determinar el tamaño del índice general de una colección, ejecute el siguiente comando:

```
db.collection.stats()
```

Para eliminar un índice no utilizado, ejecute el siguiente comando:

```
db.collection.dropIndex("indexName")
```

¿Cómo identifico los índices que faltan?

Puede utilizar el generador de perfiles de Amazon DocumentDB para registrar consultas lentas. Una consulta que aparece repetidamente en el registro de consultas lentas puede indicar que se requiere un índice adicional para mejorar el rendimiento de esa consulta.

Puede identificar las oportunidades de generar índices útiles buscando consultas de larga duración que tengan una o más etapas que realicen al menos una `COLLSCAN` etapa, lo que significa que la etapa de consulta debe leer todos los documentos de la colección para dar una respuesta a la consulta.

En el ejemplo siguiente se muestra una consulta en una colección de viajes en taxi que se ejecutaron en una colección grande.

```
db.rides.count({"fare.totalAmount":{$gt:10.0}}))
```

Para ejecutar este ejemplo, la consulta tuvo que realizar un análisis de la colección (es decir, leer cada documento de la colección), ya que no hay ningún índice en el campo `fare.totalAmount`. La salida del generador de perfiles de Amazon DocumentDB para esta consulta es similar a la siguiente:

```
{
```

```
    ...
    "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"
}
```

Para acelerar la consulta de este ejemplo, crearía un índice en `fare.totalAmount`, como se muestra a continuación.

```
db.rides.createIndex( {"fare.totalAmount": 1}, {background: true} )
```

> **Note**
>
> Los índices creados en el primer caso (es decir, cuando la opción `{background:true}` no se proporciona al crear el índice) utilizan un bloqueo de escritura exclusivo, lo que impide que las aplicaciones escriban datos en la colección hasta que se completa la creación del índice. Tenga en cuenta este impacto potencial al crear índices en clústeres de producción. Al crear índices, establézcalos. `{background:true}`

En general, le convendrá crear índices en campos que tengan una cardinalidad alta (por ejemplo, un gran número de valores únicos). Crear un índice en un campo con cardinalidad baja puede producir un índice grande que no se utilice. El optimizador de consultas de Amazon DocumentDB tiene en cuenta el tamaño general de la colección y la selectividad de los índices al crear un plan de consulta. Hay momentos en los que verá que el procesador de consultas selecciona una etapa `COLLSCAN` incluso cuando un índice está presente. Esto sucede cuando el procesador de consultas estima que la utilización del índice no ofrecerá una ventaja de rendimiento sobre el análisis de toda la colección. Si desea obligar a que el procesador de consultas utilice un índice en particular, puede utilizar el operador `hint()` como se muestra a continuación.

```
db.collection.find().hint("indexName")
```

¿Cómo determino la sobrecarga de colecciones de bases de datos?

La sobrecarga de colecciones se produce cuando el tamaño de una colección aumenta debido a la acumulación de documentos inactivos u obsoletos o a la fragmentación de las páginas de la base de datos. El porcentaje informado representa la cantidad de espacio de documento que pueden utilizar documentos futuros. Esta sobrecarga consume espacio tanto en la memoria caché como en el almacenamiento del búfer. Para eliminar la sobrecarga, las colecciones deben volver a cargarse mediante un bucle de migración dump/restore o mediante un bucle de migración y un cambio durante un período de mantenimiento.

Example Ejemplo

Ejecute el siguiente comando para determinar el almacenamiento no utilizado de la colección:

```
db.runCommand({collStats:'coll'})
```

El resultado tendrá un aspecto parecido al siguiente:

```
{
        "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)
}
```

Resumen de consultas útiles

Las siguientes consultas pueden resultar útiles para monitorizar el rendimiento y el uso de recursos en Amazon DocumentDB..

- Utilice el siguiente comando para ver las estadísticas de una colección específica, incluidos los contadores de operaciones, las estadísticas de memoria caché, las estadísticas de acceso y las estadísticas de tamaño:

  ```
  db.collection.stats()
  ```

- Utilice el siguiente comando para ver las estadísticas de cada índice creado en una colección, incluido el tamaño del índice, las estadísticas de caché específicas del índice y las estadísticas de uso del índice:

  ```
  db.collection.aggregate([{$indexStats:{}}]).pretty()
  ```

- Utilice la siguiente consulta para ver toda la actividad.

  ```
  db.adminCommand({currentOp: 1, $all: 1});
  ```

- El código siguiente muestra una lista de todas las consultas de ejecución prolongada o bloqueadas.

```
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: {}
                });
```

- El código siguiente finaliza una consulta.

```
db.adminCommand({killOp: 1, op: <opid of running or blocked query>});
```

- Utilice el código siguiente para obtener una vista acumulada del estado del sistema.

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
                            {$group: {_id: {desc: "$desc", ns: "$ns", WaitState:
 "$WaitState"}, count: {$sum: 1}}}],
                 cursor: {}
                });
```

Recopilación de elementos no utilizados en Amazon DocumentDB

Amazon DocumentDB implementa una arquitectura de base de datos de control de concurrencia multiversión (MVCC) que crea nuevas versiones de entradas de documentos e índices para cada operación de actualización. Esta arquitectura permite el aislamiento de las transacciones, lo que evita que los cambios de una transacción aparezcan en otra.

Temas

- Descripción de la recolección de basura en Amazon DocumentDB
- Proceso de recopilación de elementos no utilizados
- Arquitectura de almacenamiento y almacenamiento extendido
- Supervisión de recopilación de elementos no utilizados

- Ejemplo de salida de CollStats
- Preguntas frecuentes

Descripción de la recolección de basura en Amazon DocumentDB

La recopilación de elementos no utilizados (GC) es un proceso en segundo plano automatizado que mantiene un rendimiento y una disponibilidad óptimos del sistema en Amazon DocumentDB. Al igual que muchas bases de datos modernas, la arquitectura MVCC de Amazon DocumentDB crea nuevas versiones de documentos e índices con cada actualización. Cada operación de escritura consume un identificador MVCC único de un contador finito. Estos ID identifican a qué transacción pertenece una versión del documento y si se ha confirmado o anulado. Con el tiempo, estas versiones antiguas y sus ID de MVCC se acumulan, por lo que es necesario limpiarlas para evitar una degradación del rendimiento.

Funciones de la recopilación de elementos no utilizados

El recopilador de elementos no utilizados cumple tres funciones esenciales:

- Recupera espacio de almacenamiento: elimina las versiones obsoletas de documentos e índices que las consultas activas ya no necesitan, lo que libera espacio para futuras operaciones de escritura.
- Evita el desbordamiento de identificadores de MVCC: evita el desbordamiento de identificadores de MVCC mediante la administración del contador finito de identificadores de MVCC. Sin esta administración, el contador acabaría por alcanzar su límite, lo que obligaría a la base de datos a pasar a un modo temporal de solo lectura hasta que se reciclen los identificadores.
- Mantiene el rendimiento de las consultas: mantiene un rendimiento óptimo de las consultas, ya que elimina las versiones inactivas de los documentos que, de otro modo, se acumularían y ralentizarían el procesamiento de las consultas.

Proceso de recopilación de elementos no utilizados

El proceso de GC funciona por colección y puede tener varios procesos ejecutándose simultáneamente en diferentes colecciones. El proceso consta de cuatro fases secuenciales:

1. Identificación: el sistema identifica las versiones de documentos e índices a las que ya no hacen referencia las transacciones ni consultas activas.

2. Carga de memoria: los documentos antiguos y las entradas del índice se cargan en la memoria si aún no están presentes.
3. Eliminación: las versiones obsoletas se eliminan permanentemente para recuperar espacio de almacenamiento.
4. Reciclaje de identificadores de MVCC: el sistema recicla los identificadores de MVCC de las versiones eliminadas para realizar nuevas operaciones.

Cuando la recopilación de elementos no utilizados termina de procesar las versiones antiguas de los documentos, elimina del sistema los identificadores del MVCC más antiguos. Esta limpieza es crucial para evitar el desbordamiento de los identificadores de MVCC, ya que se reciclan los identificadores de MVCC y los pone a disposición para nuevas operaciones de escritura en todo el clúster. Sin este proceso de reciclaje, el sistema acabaría agotando su contador finito de identificadores de MVCC y pasaría a un estado de solo lectura.

Programación de recopilación de elementos no utilizados

La recopilación de elementos no utilizados se ejecuta automáticamente en segundo plano a intervalos periódicos. La temporización y la frecuencia se ajustan dinámicamente en función de la carga del sistema, los recursos disponibles, el volumen de escritura y los niveles de consumo de los identificadores de MVCC. Cuando hay mucha actividad de escritura, el proceso de GC se ejecuta con más frecuencia para administrar el aumento del número de versiones de los documentos.

Arquitectura de almacenamiento y almacenamiento extendido

Amazon DocumentDB utiliza una arquitectura de almacenamiento sofisticada que separa el almacenamiento de documentos en dos segmentos distintos:

Segmento de almacenamiento base

El segmento de almacenamiento base contiene los datos y metadatos del documento principal. Este segmento almacena:

- Contenido del documento que se ajusta al tamaño de página estándar (8 KB).
- Metadatos del documento e información de estructura.
- Índices principales y sus entradas.
- Collection-level estadísticas y configuración.

Segmento de almacenamiento extendido

El segmento de almacenamiento extendido utiliza un almacén especializado de objetos de documentos de gran tamaño diseñado para gestionar documentos que superan el tamaño de página de almacenamiento estándar. Este segmento proporciona:

- Gestión eficiente de documentos de gran tamaño: los documentos que superan el umbral de almacenamiento básico se mueven automáticamente al segmento de almacenamiento extendido.
- Diseño de almacenamiento optimizado: el segmento utiliza un formato de almacenamiento diferente optimizado para objetos grandes, lo que reduce la fragmentación y mejora los patrones de acceso.
- Recolección de basura independiente: el segmento de almacenamiento extendido tiene su propio proceso de recolección de basura que puede ejecutarse independientemente de la limpieza del almacenamiento base.
- Acceso transparente: las aplicaciones acceden a documentos grandes sin problemas sin necesidad de saber qué segmento de almacenamiento contiene los datos.

El segmento de almacenamiento extendido es particularmente beneficioso para:

- Colecciones con documentos que contienen matrices incrustadas de gran tamaño.
- Documentos con amplias estructuras anidadas.
- Colecciones que almacenan datos binarios o campos de texto de gran tamaño.
- Aplicaciones con tamaños de documentos mixtos en las que algunos documentos superan con creces el tamaño promedio.

Supervisión de recopilación de elementos no utilizados

Métricas de nivel de clúster

`AvailableMVCCIds`

- Ubicación: Amazon CloudWatch
- Descripción: contador que muestra el número de operaciones de escritura restantes disponibles hasta un límite máximo de 1800 millones. Cuando este contador llega a cero, el clúster pasa al modo de solo lectura hasta que se recuperen y reciclen los identificadores. El contador disminuye

con cada operación de escritura y aumenta a medida que la recopilación de elementos no utilizados recicla los identificadores de MVCC antiguos.

- Recomendación: configure una alarma cuando el valor caiga por debajo de los 900 millones en Amazon DocumentDB 5.0 y versiones posteriores, o por debajo de los 1300 millones en Amazon DocumentDB 3.6 y 4.0. Esta alerta temprana le permite tomar las medidas recomendadas que se describen más adelante.

`LongestActiveGCRuntime`

- Ubicación: Amazon CloudWatch
- Descripción: duración en segundos del proceso de recopilación de elementos no utilizados más prolongado. Se actualiza cada minuto y realiza un seguimiento únicamente de las operaciones activas, excluyendo los procesos que se completan en el intervalo de un minuto.
- Recomendación: compárelos con los datos `gcRuntimeStats` históricos para identificar un comportamiento anormal en la recolección de basura, como la prolongación de los tiempos de ejecución durante las eliminaciones masivas.

Métricas a nivel de la colección

`MVCCIDStats: MVCCIdScale`

- Ubicación: comando collStats de la base de datos
- Descripción: mide la antigüedad del identificador de MVCC en una escala de 0 a 1, donde 1 indica la antigüedad máxima antes de que un clúster entre en estado de solo lectura. Utilice esta métrica junto con `AvailableMVCCIds` para identificar las colecciones que contienen los identificadores de MVCC más antiguos que están deteriorando el clúster.
- Recomendación: mantenga los valores por debajo de 0,3 para cada colección.

`gcRuntimeStats`

- Ubicación: comando collStats de la base de datos
- Descripción: proporciona un historial de dos meses con las métricas de recopilación de elementos no utilizados, que incluye el total de ejecuciones, la duración promedio y la duración máxima. Solo incluye las operaciones de recopilación de elementos no utilizados que duren más de cinco minutos para garantizar estadísticas significativas.

`storageSizeStats`

- Ubicación: comando collStats de la base de datos
- Descripción: proporciona un desglose detallado de la utilización del almacenamiento en los diferentes segmentos de almacenamiento:
  - `storageSegmentBase`— Almacenamiento utilizado por el segmento de almacenamiento base para documentos estándar
  - `storageSegmentExtended`— Almacenamiento utilizado por el segmento de almacenamiento extendido para documentos de gran tamaño
- Uso: ayuda a identificar las colecciones con un almacenamiento significativo de documentos de gran tamaño y a comprender los patrones de distribución del almacenamiento.

`unusedStorageSize` (nivel de colección)

- Ubicación: comando collStats de la base de datos
- Descripción: estima el espacio de almacenamiento no utilizado en una colección en función de las estadísticas muestreadas. Incluye el espacio de los documentos eliminados y de los segmentos vacíos. La métrica proporciona los totales combinados y los desgloses por segmento:
  - Combinados `unusedBytes` y en `unusedPercent` todos los segmentos de almacenamiento
  - `storageSegmentBase`— Espacio no utilizado específicamente en el segmento de almacenamiento base
  - `storageSegmentExtended`— Espacio no utilizado específicamente en el segmento de almacenamiento extendido

`documentFragmentStats`

- Ubicación: comando collStats de la base de datos
- Descripción: proporciona información detallada sobre los fragmentos de documentos y los datos inactivos de las colecciones. Los fragmentos de documentos representan las unidades de almacenamiento interno que utiliza el motor de base de datos, y los fragmentos inactivos indican datos a los que ya no se puede acceder pero que aún no se han recuperado. Esta métrica incluye:
  - `totalDocFragmentsCount`— Número total de fragmentos de documentos de la colección
  - `deadDocFragmentsCount`— Número de fragmentos que contienen datos muertos (inaccesibles)

- `deadDocFragmentsPercent`— Porcentaje de fragmentos que contienen datos inactivos
- `deadDocFragmentBytes`— Estimación de bytes consumidos por fragmentos de documentos inactivos
- Per-segment desglose por `storageSegmentBase` y `storageSegmentExtended`

- Uso: supervise esta métrica para comprender la eficacia de la recolección de basura e identificar las recolecciones que podrían beneficiarse de las operaciones de mantenimiento. Los porcentajes altos de fragmentos muertos indican que la recolección de basura puede estar retrasándose o que la recolección se beneficiaría de una optimización.

Métricas de nivel de índice

`unusedStorageSize` (nivel de índice)

- Ubicación: comando indexStats de la base de datos.
- Descripción: estima el espacio de almacenamiento no utilizado en un índice en función de las estadísticas muestreadas. Incluye el espacio de las entradas de índice obsoletas y de los segmentos vacíos.
- Recomendación: utilice el comando `reIndex` para compilar los índices sin tiempo de inactividad y recuperar el espacio no utilizado. Consulte Administración de índices para obtener más detalles.

Ejemplo de salida de CollStats

El siguiente ejemplo muestra un `collStats` resultado típico con métricas de recolección y almacenamiento de basura:

```
{
    "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)
}
```

Preguntas frecuentes

¿Cómo puedo identificar si la recopilación de elementos no utilizados no funciona de manera eficiente?

Supervise estas señales de advertencia que indican una recopilación de elementos no utilizados ineficiente:

- Excesiva acumulación de datos: `unusedStorageSize` las métricas aumentan de forma constante durante las tareas de escritura intensiva o las eliminaciones masivas, especialmente con índices de gran tamaño.
- Porcentaje elevado de fragmentos muertos: `documentFragmentStats` muestra `deadDocFragmentsPercent` valores altos de forma constante (por encima del 10 al 15%).
- Latencia de consulta degradada: aumento de la latencia de consulta debido a la acumulación de documentos muertos.
- Duración prolongada de la GC: las operaciones de recolección de basura tardan más que los promedios históricos en. `gcRuntimeStats`
- Procesamiento de GC elevado: alto, `LongestActiveGCRuntime` lo que indica que el recolector de basura no puede satisfacer las demandas del sistema.

¿La recopilación de elementos no utilizados afecta al rendimiento de mi base de datos?

En condiciones normales, la recopilación de elementos no utilizados tiene un impacto mínimo en el rendimiento. Sin embargo, si la recopilación de elementos no utilizados se retrasa, puede experimentar lo siguiente:

- Mayores costos de almacenamiento debido a la acumulación de documentos muertos.
- El rendimiento de las consultas es más lento debido a que las entradas de índice están obsoletas.
- Modo temporal de solo lectura si se agotan los ID de MVCC.
- Mayor uso de recursos durante las ejecuciones de recopilación intensivas, especialmente en instancias más pequeñas.
- Reducción de la eficiencia en las operaciones del segmento de almacenamiento extendido para documentos de gran tamaño.

¿Puedo activar manualmente la recopilación de elementos no utilizados?

No, la recopilación de elementos no utilizados en Amazon DocumentDB no se puede activar manualmente. El sistema administra la recopilación de elementos no utilizados automáticamente como parte de sus operaciones de mantenimiento interno.

¿Qué alarmas debo configurar como práctica operativa recomendada?

Configure la supervisión tanto a nivel de clúster como de recopilación para garantizar un rendimiento óptimo de su sistema Amazon DocumentDB.

Para la supervisión a nivel de clúster, comience por crear una CloudWatch alarma de Amazon para la `AvailableMVCCIds` métrica con un umbral de 900 millones en Amazon DocumentDB 5.0 y versiones posteriores, o de 1300 millones en Amazon DocumentDB 3.6 y 4.0. Esto le da tiempo suficiente para tomar medidas antes de que la métrica llegue a cero, momento en el que el clúster pasará al modo de solo lectura. Tenga en cuenta que esta métrica puede variar en función de sus patrones de uso específicos: algunos clientes ven que cae por debajo del umbral y, luego, se recupera por encima de los 1500 millones a medida que la recolección de basura completa su trabajo.

También es importante supervisar la `LongestActiveGCRuntime` métrica a través de Amazon CloudWatch. Esta métrica, junto con `gcRuntimeStats`, le permite comprender el rendimiento de la recopilación de elementos no utilizados en todo el sistema.

Para la supervisión a nivel de recopilación, céntrate en estas métricas clave:

- `MVCCIdScale`— Esté atento a los valores crecientes que sugieren que los ID de MVCC están envejeciendo y pueden necesitar atención.
- `gcRuntimeStats`— Identifique los procesos de recolección de basura que tardan de manera inusual o se extienden durante varios días.
- `documentFragmentStats`— Controle `deadDocFragmentsPercent` los valores: los porcentajes consistentemente altos (por encima del 10 al 15%) pueden indicar que la recolección de basura se está retrasando.
- `storageSizeStats`y `unusedStorageSize` — Realice un seguimiento de los patrones de utilización del almacenamiento e identifique las colecciones con una cantidad significativa de espacio no utilizado en cualquiera de los segmentos de almacenamiento.

Las colecciones con operaciones de escritura frecuentes requieren atención especial, ya que generan más trabajo para el recopilador de elementos no utilizados. Compruebe estas métricas con más frecuencia para ver si hay recopilaciones con mucha actividad de escritura, a fin de garantizar que la recolección de basura esté a la altura de su carga de trabajo.

Tenga en cuenta que estas recomendaciones de supervisión sirven de punto de partida. A medida que se familiarice con el comportamiento de su sistema, tal vez desee ajustar estos umbrales para que se adapten mejor a sus patrones y requisitos de uso específicos.

¿Qué debo hacer si mis **`MVCCID disponibles están por`** debajo del umbral de alarma?

Si su `AvailableMVCCIds` métrica cae por debajo del umbral de alarma, tome medidas de inmediato para evitar que el clúster entre en modo de solo lectura. En primer lugar, amplía el tamaño de la instancia para proporcionar más recursos informáticos al recolector de basura. Esto permite que su aplicación continúe con sus operaciones normales y, al mismo tiempo, proporciona la potencia adicional que necesita el recopilador de elementos no utilizados para ponerse al día.

Si la ampliación por sí sola no mejora la situación, considera la posibilidad de reducir las operaciones de escritura. Use la métrica `MVCCIdScale` para identificar qué colecciones específicas contienen identificadores de MVCC antiguos que requieren atención. Además, supervise `documentFragmentStats` para identificar las colecciones con altos porcentajes de fragmentos muertos que podrían estar contribuyendo a la ineficiencia de la recolección de basura.

Una vez que hayas identificado estas colecciones, es posible que tengas que reducir temporalmente las operaciones de escritura en ellas para que la recolección de basura se ponga al día. Durante el período de recuperación, supervisa de cerca la `AvailableMVCCIds` métrica para asegurarte de que tus acciones tengan el efecto deseado. Se considera que el clúster está en buen estado una vez que el valor `AvailableMVCCIds` vuelve a ser 1500 millones o más.

Recuerde que estos pasos son medidas preventivas para ayudar a que su sistema se recupere antes de que alcance un estado crítico. Cuanto antes actúes después de ver que la métrica cae por debajo del umbral, más probabilidades tendrás de evitar cualquier impacto en tus operaciones de escritura.

¿Las operaciones por lotes afectan a la tasa de consumo de los ID de MVCC?

Sí. Como un lote ejecutado dentro de una sola transacción atómica consume solo un ID de MVCC, al agrupar las escrituras en lotes se reduce la velocidad a la que se consumen los ID de MVCC. Como práctica recomendada, utilice `insertMany` y `updateMany` siempre que sea posible para minimizar el consumo de ID de MVCC y reducir la carga de recolección de basura.

¿Por qué los **`MVCCID disponibles no se`** recuperan hasta que alcanzan un determinado umbral?

Se espera que una carga de trabajo que predomine la escritura y que aún no haya consumido suficientes identificadores de MVCC como para iniciar el ciclo de recuperación de basura, disminuya de forma constante y sin recuperación. `AvailableMVCCIds` La recolección de basura se activa cuando se cumple alguna de las siguientes condiciones: una parte determinada de los documentos de una colección queda obsoleta, una parte determinada de los documentos nuevos se inserta o `AvailableMVCCIds` cae por debajo del umbral de recuperación.

La recuperación forzosa no se produce hasta que esté por `AvailableMVCCIds` debajo de su umbral de recuperación. Una vez que lo haga, la recuperación se activará y `AvailableMVCCIds` se recuperará. Sin embargo, las transacciones de larga duración y los cursores abiertos pueden bloquear el horizonte de recuperación e impedir la recuperación incluso después de alcanzar el umbral.

Códigos de error

Cuando se produce un error en un comando, recibe un error de Amazon DocumentDB (compatible con MongoDB) que contiene un error numérico `code` y un mensaje descriptivo (). `errmsg` Utilice la siguiente tabla para identificar los códigos de error que Amazon DocumentDB 8.0 podría

devolver a su aplicación cliente. Cuando un código de error corresponde a una sola condición, la descripción coincide con el texto del mensaje devuelto por el servidor. Los códigos que cubren varias condiciones tienen una descripción general.

> **Important**
>
> No cree una lógica de aplicación que dependa de códigos de error numéricos específicos, ya que Amazon DocumentDB puede cambiar los códigos de error entre las versiones principales.

Código de error	Description (Descripción)
1	No se encontró el artículo
2	El argumento, la opción o el valor de un campo de un comando no es válido o no se admite.
4	Falta un campo obligatorio o un valor de configuración no tiene el formato correcto.
5	Una definición de vista hace referencia a sí misma, creando un ciclo o se ha producido un error en el sistema de archivos.
9	No se pudo analizar un comando, campo, sugerencia o símbolo de reanudación o su formato no es válido.
10	No se encontró la colección de destino o no se especificó ninguna colección.
11	No se encontró un usuario al que se hace referencia o la colección de destino ya existe.
13	No está autorizado a ejecutar este comando o su autenticación no es válida.
14	Un campo o valor tiene un tipo de datos incorrecto.
15	Un valor sobrepasó el rango permitido durante la evaluación.

Código de error	Description (Descripción)
16	El lote de escritura está vacío, tiene un formato incorrecto o supera el recuento de operaciones permitido.
17	No todos los objetos son del mismo tipo
18	La autenticación falló porque las credenciales no son válidas o el usuario no existe.
19	No se encontró la operación primitiva de procesamiento de datos o no se encontró el tipo primitivo
20	La operación solicitada no está permitida en este contexto.
26	El espacio de nombres o la colección de destino no existen.
27	El índice solicitado no existe o no hay ningún índice adecuado disponible.
37	Falta un argumento de canalización o etapa de agregación, está vacío o no es válido.
38	No se encontró el nombre de base de datos
39	La operación se canceló porque superó el límite máximo de tiempos, duró demasiado o superó la memoria disponible.
40	Una actualización entra en conflicto con otro campo o la recolección de basura limita o deshabilita las escrituras.
41	JavaScript error
43	El cursor especificado ya no existe o ha caducado.
44	El índice de matriz está fuera de los límites
48	Ya existe una colección, una vista o un espacio de nombres con ese nombre.
51	Operador de actualización ilegal

Código de error	Description (Descripción)
52	Un nombre de campo usa un prefijo '$' reservado que no es válido para su almacenamiento.
53	El campo _id tiene un tipo no válido o no admitido, como una matriz o una expresión regular.
54	no se pueden deducir los campos de consulta que se van a establecer
55	La transacción ya se ha confirmado
56	El nombre de un campo de la solicitud está vacío, lo que no está permitido.
57	Transacción invalidada por otra transacción
58	El valor del campo es demasiado largo
59	El comando solicitado es desconocido o no se admite.
61	Recopilación no válida
62	No se pueden realizar cambios en la colección
65	Se han producido errores de operación por lotes
66	Una actualización intentó modificar el campo _id inmutable.
67	No se pudo crear el índice debido a una especificación no válida o no admitida.
68	Ya existe un índice con propiedades conflictivas.
70	el campo usado con $snippet no está configurado como columna de fragmentos
71	El índice ya existe con diferentes opciones
72	Una o más opciones especificadas para el comando o el índice no son válidas.
73	El nombre del espacio de nombres, la base de datos o la colección no es válido o no está permitido para esta operación.

Código de error	Description (Descripción)
74	La colección no tiene un índice de texto definido.
75	Se ha definido un índice de texto para esta colección, pero no para este campo.
77	No se puede usar la instantánea con la clasificación
78	No se encuentra el documento
80	La consulta supera el tamaño máximo permitido.
82	El formato del marco temporal no es válido. <number>Formato válido: «1 m» o «3» '//' h/d w/s m/y
85	Ya existe un índice con ese nombre, pero con diferentes opciones.
86	El índice solicitado entra en conflicto con un índice existente que tiene el mismo nombre.
87	El orden de las claves $sort debe ser 1 (para el orden ascendente) o -1 (para el orden descendente)
92	No hay suficiente espacio para operar
93	El usuario ya existe
94	El privilegio de recopilación, clúster o recurso de destino no es válido o no está disponible.
95	La base de datos no existe
96	El parámetro 'writeConcern' no es válido
97	La contraseña no puede estar vacía
99	Función o privilegio indefinido
102	No se puede agregar un rol que ya está integrado

Código de error	Description (Descripción)
103	No se pudo extraer un rol válido de la colección.
105	El usuario no es válido o no se pudo inicializar la biblioteca de autorizaciones.
107	No se pueden asignar funciones a la base de datos desde otra base de datos
111	_id debe especificarse exactamente al proporcionar el documento de actualización
112	Escribe Conflicto
113	El rol especificado no existe.
114	El rol ya existe
115	AnyResource debe ser verdadero cuando se especifica
116	los recursos deben tener un conjunto de bases de datos y colecciones, un conjunto de clústeres o un conjunto de recursos
117	La operación entra en conflicto con otra operación en curso, como una transacción activa.
120	No se puede especificar 'ShowPrivileges' porque se muestra a todos los usuarios
121	No se pudo validar el documento
122	Error de actualización
124	Error al procesar el identificador del esquema
125	El comando falló, a menudo debido a una referencia ambigua o a la falta de sesión.
126	La base de datos ya existe
127	El nombre de la base de datos ya existe en mayúsculas y minúsculas

Código de error	Description (Descripción)
128	Demasiados campos
129	Esperando eliminar la base de datos
130	Valor de parámetro incorrecto
136	Se perdió la posición del cursor o la función de flujo de cambios requerida no está habilitada.
139	Fallo de JS Interpreter
145	El alias no está definido en la consulta
147	no maestro
148	Recursos insuficientes para ejecutar la consulta
149	El alias ya está definido en la consulta
154	Un argumento tiene un valor no válido o un tipo incorrecto (no numérico).
155	Se encontró un valor NULO
156	Se encontró un valor INDEFINIDO
157	Se encontró un valor NAN
158	La colección ya está en uso para otro propósito
159	Se ha desbordado un valor de duración o se ha producido un error en una operación de red.
161	La versión de intercalación solicitada no es válida o no está disponible.
163	Un campo o un nombre no son válidos para su almacenamiento.
164	Un operador de consulta se anidó de una forma que no está permitida.

Código de error	Description (Descripción)
165	Una vista supera la profundidad máxima de anidación de 20 o hace referencia a sí misma en un ciclo.
166	La operación no se admite en una vista o se han utilizado demasiados elementos posicionales.
167	La operación o etapa solicitada no es compatible con una vista.
168	No se reconoce una expresión o un operador de canalización o tiene un argumento no válido.
169	BadValue $maxTimeMS tiene un valor no integral
170	BadValue $maxTimeMS está fuera de rango
171	No se puede crear un índice sobre varios campos de matriz.
173	El valor de omisión debe ser positivo
174	La verbosidad esperada del registro debe ser un número
181	El patrón de claves coincide con varios índices; en su lugar, especifique el índice por su nombre.
197	La especificación del índice contiene un campo no válido o no admitido.
207	uuid debe ser un campo binario de 16 bytes con el subtipo UUID (4)
224	Esta función de consulta no está permitida en el contexto actual.
225	Se inició una transacción más reciente en la sesión, por lo que la anterior no puede continuar.
241	No se pudo convertir un valor al tipo solicitado en $convert.
250	Vista no válida

Código de error	Description (Descripción)
251	El número de transacción especificado no coincide con ninguna transacción activa.
252	Se ha alcanzado el límite de recursión
253	No se pudo insertar la entrada en la vista
254	No se pudo eliminar la entrada de la vista
256	Documento filtrado por la cláusula $having
257	El tamaño total de todas las operaciones de la transacción supera el límite permitido.
263	Esta operación no se puede ejecutar dentro de una transacción de varios documentos.
291	no se puede encontrar el índice para la consulta $geoNear
300	La autenticación falló debido a un protocolo de enlace de SASL o a un error de sesión.
301	No se admite ningún mecanismo de autenticación ni una operación solicitada.
302	La autenticación necesita datos adicionales para completar el intercambio de SASL.
303	No se admite la configuración de etapa, opción o rol solicitada.
304	No se admite la etapa de agregación solicitada.
305	No se admite el operador de proyección de agregación.
306	No se admite el operador acumulador del grupo de agregación especificado.
502	La columna ya existe
505	Aquí no se permite la inserción

Código de error	Description (Descripción)
601	Las coordenadas están fuera de los límites
602	Error de índice
700	No se permiten cambios en este parámetro en el modo en vivo
701	Valor no válido para este parámetro
703	Transición de estado ilegal para este parámetro
999	ID de transacción incorrecto
4570	El parámetro $lookup espera el valor de la cadena
4572	A $lookup le faltan los campos obligatorios
10065	A un parámetro que debe ser un objeto (como vars o $rand) se le asignó un valor que no era objeto.
10107	no maestro
11000	Una clave duplicada infringe una restricción de índice única.
11601	La operación se interrumpió.
11603	La solicitud del cliente está dañada o su sintaxis es ilegal
13103	demasiadas claves compuestas
13297	Ya existe una base de datos con el mismo nombre, pero solo difiere en mayúsculas y minúsculas.
13388	Nombre de fase del oleoducto no reconocido: $CollStats
13435	no master y slaveOK=false
14031	No hay suficiente espacio en disco
15888	debe pasar el nombre de la colección para crearla

Código de error	Description (Descripción)
15947	los campos de un grupo deben especificarse en un objeto
15950	El nombre del campo agregado del grupo no puede ser un operador
15952	operador $group desconocido
15955	Error de sintaxis en el operador de agregación de grupos: se debe especificar el campo _id
15956	El argumento $skip es negativo; debe ser un número no negativo.
15957	el límite debe especificarse como un número
15958	El valor del $limit debe ser un número positivo.
15959	el filtro de coincidencia debe ser una expresión de un objeto
15969	Error de sintaxis: el valor del operador $project debe ser un documento.
15972	El argumento para $skip debe ser un número
15973	la especificación de la clave $sort debe ser un objeto
15974	$sort key: el orden debe especificarse mediante un número
15975	El orden de las claves $sort debe ser 1 (para el orden ascendente) o -1 (para el orden descendente)
15976	$sort stage debe tener al menos una clave de clasificación
15981	el campo de consulta no es una cadena
15983	Un objeto que represente una expresión debe tener exactamente un campo.
15992	tipo de campo no permitido: NULL en la expresión de objeto
15998	Los nombres de las rutas de campo no pueden ser cadenas vacías.
16006	No se pudo convertir un valor en una fecha porque no es un tipo de fecha válido.

Código de error	Description (Descripción)
16007	No se puede convertir un valor en una cadena porque se esperaba una cadena.
16020	Se llamó a un operador de expresión con un número incorrecto de argumentos.
16034	El índice inicial de $SubstrBytes debe ser de tipo numérico.
16035	El argumento de longitud $substrBytes debe ser de tipo numérico.
16395	$where no está permitido dentro de una expresión de agregación de $match
16400	no puede agregar una expresión para un campo porque ya existe una expresión para ese campo o uno de sus subcampos.
16401	no puede agregar una expresión para un subcampo porque ya hay una expresión que se aplica a todo el campo
16404	Las expresiones $ no están permitidas en este contexto
16406	No se pueden excluir campos distintos de _id
16410	El nombre de un campo o una ruta de $project comienzan ilegalmente con «$» o lo contienen.
16411	Los nombres de campo no pueden contener caracteres nulos
16419	la ruta del campo no debe contener caracteres nulos incrustados
16424	$near no está permitido dentro de una expresión de agregación de $match
16426	No se permite usar $nearSphere/$near dentro de una expresión de agregación de $match
16435	El componente de canalización debe contener un elemento
16493	Intenté crear una cadena de más de 16 MB
16554	$add solo admite tipos numéricos o de fecha.

Código de error	Description (Descripción)
16605	$GeoNear requiere un campo «cercano» válido y opciones de GeoNear escritas correctamente.
16606	$geoNear requiere la opción 'DistanceField' como cadena
16607	$geoNear requiere que la opción dada se especifique como una cadena.
16608	Divide por cero
16610	A $mod se le dio un divisor de cero.
16611	$mod solo admite tipos numéricos para ambos operandos.
16612	solo se permite una fecha en una expresión $add
16613	La operación intentó restar de una fecha un valor no numérico ni de fecha.
16702	$concat solo admite argumentos de cadena.
16730	el texto IndexVersion está fuera de rango
16755	no puede indexar la geometría con un orden de bobinado estricto
16800	no puede tener 2 campos geográficos
16801	2d tiene que ser el primero en el índice
16804	objeto de ubicación esperado, la matriz de ubicación no tiene el formato correcto
16810	$mod requiere una matriz de exactamente dos valores como parámetro.
16837	Una ruta de actualización no puede modificar el mismo campo más de una vez.
16868	Nombre de variable no válido
16869	Nombre de variable no válido
16870	Nombre de variable no válido

Código de error	Description (Descripción)
16872	Una ruta de campo está vacía o solo se usa «$» como ruta de campo en $project.
16873	Fieldpath no comienza con $
16874	$let solo admite un objeto como argumento
16875	Se le asignó un parámetro no reconocido a $let.
16876	Falta el parámetro 'vars' en $let
16877	Falta el parámetro 'in' en $let
16878	$map solo admite un objeto como argumento
16879	$map recibe un parámetro no reconocido.
16880	A $map le falta el parámetro de «entrada» requerido.
16881	Falta un campo obligatorio
16882	Falta el parámetro 'in' en $map
16883	La entrada a $map debe ser una matriz.
16898	$minDistance no debe ser negativo
16899	$maxDistance debe ser un número
16900	$maxDistance no debe ser negativo
16945	Se ha superado el límite de memoria de $group o $lookup. El pase permite DiskUse:true registrarse.
16949	Opción de agregación no válida; allow DiskUse debe ser un valor booleano.
16990	$out requiere un argumento de cadena u objeto que nombre el objetivo de salida.
16996	error al insertar para $out

Código de error	Description (Descripción)
16997	$out falló porque los índices de la colección de destino cambiaron durante el procesamiento.
17040	ElementsTrueEl argumento de $all debe ser una matriz.
17041	ElementTrueEl argumento de $any debe ser una matriz.
17042	Ambos operandos de $set IsSubset deben ser matrices.
17043	Todos los operandos de $setUnion deben ser matrices.
17044	Un operando de $setEquals no es una matriz.
17045	$setEquals requiere al menos dos argumentos.
17046	Ambos operandos de $set IsSubset deben ser matrices.
17047	Todos los operandos de $setIntersection deben ser matrices.
17048	Ambos operandos de $setDifference deben ser matrices.
17049	Ambos operandos de $setDifference deben ser matrices.
17053	La expresión de $redact no debe devolver nada aparte de las variables $$KEEP, $$DESCEND y $$PRUNE
17080	Falta el parámetro 'if' para cond
17081	Falta el parámetro «entonces» para cond
17082	Falta el parámetro 'else' para configurar
17083	Parámetro no reconocido para $cond
17124	El argumento de $size debe ser una matriz.
17275	No se puede redefinir ROOT
17276	No se encontró la variable

Código de error	Description (Descripción)
17280	El valor de la clave de índice es demasiado grande para indexarlo.
17287	$merge no puede modificar la misma fila de destino más de una vez.
17293	la opción de índice de texto 'textIndexVersion' debe ser un número
17302	El valor de este argumento debe ser mayor o igual a cero.
17304	$geoNear requiere argumentos de consulta geográfica y de punto «cercano» válidos.
17308	$meta recibe un argumento sin fundamento.
17310	Ambos operandos de $set IsSubset deben ser matrices.
17313	$match with $text solo está permitido como primera etapa del proceso.
17320	CreateCollection no se puede ejecutar en un espacio de nombres que contenga un nombre reservado.
17385	No se puede destinar dinero a una colección especial
17390	$group no admite expresiones de estilo inclusivo
17419	Tras la actualización, el documento resultante supera el tamaño máximo permitido.
17420	El documento a alterar es mayor que 16777216
18533	El campo de formato debe ser una cadena
18534	Campo no reconocido hasta $date ToString
18535	La cadena con formato de fecha no es válida debido a un signo de porcentaje al final.
18536	Formato $date time ToString no válido

Código de error	Description (Descripción)
18537	$date ToString no puede formatear un componente de fecha que esté fuera del rango 0-9999.
18627	Falta el parámetro de «formato» para $date ToString
18628	Falta el parámetro 'date' para $date ToString
18629	tipo de argumento incorrecto
25278	$geoNear requiere un argumento 'DistanceField'
26823	No se puede representar un identificador de operación dentro de los bits disponibles.
28539	El primer argumento del comando distinto debe ser una cadena
28646	$Filter espera un objeto como parámetro
28647	Campo inesperado para $filter
28648	$Filter requiere un campo de «entrada»
28649	$Filter requiere un campo «como»
28650	$Filter requiere un campo «cond»
28651	Se requiere una entrada de matriz
28656	$substrBytes: rango no válido, el índice inicial es un UTF-8 byte de continuación.
28657	$substrBytes: rango no válido, el índice final está en medio de un carácter. UTF-8
28664	$concatArrays solo admite entradas de matriz.
28667	Una expresión como $slice, $range o $index OfArray tiene un número incorrecto de argumentos.

Código de error	Description (Descripción)
28680	$abs se aplica a un valor, como el valor mínimo de longitud, que no se puede negar.
28689	El argumento de un operador debe ser una matriz, pero se especificó un tipo diferente.
28690	El segundo argumento de la matriz ElemAt debe ser numérico
28691	El segundo argumento de la matriz ElemAt debe estar representado por un entero de 32 bits
28714	El argumento de $sqrt debe ser mayor o igual a 0
28724	El primer argumento de $slice debe ser una matriz.
28725	El segundo argumento de $slice debe ser un valor numérico.
28726	El segundo argumento de $slice debe caber en un entero de 32 bits.
28727	El tercer argumento de $slice debe ser un número.
28728	El tercer argumento de $slice debe ser un entero de 32 bits.
28729	El tercer argumento de $slice debe ser un entero positivo de 32 bits.
28745	la especificación $sample stage debe ser un objeto
28746	el argumento de tamaño para $sample debe ser un número
28747	el argumento de tamaño para $sample no debe ser negativo
28748	$sample recibió una opción no reconocida.
28749	$sample stage debe especificar un tamaño
28756	El argumento de $log debe ser numérico.
28757	$log requiere que su argumento base sea un valor numérico.

Código de error	Description (Descripción)
28758	El argumento de $log debe ser un número positivo.
28759	la base de $log debe ser un número positivo no igual a 1.
28761	El argumento de $log10 debe ser un número positivo.
28762	La base de $pow debe ser un valor numérico.
28763	El exponente de $pow debe ser numérico.
28764	$pow no puede tomar una base de 0 y un exponente negativo
28765	$ceil, $exp, $floor, $ln o $log10 reciben un tipo no numérico.
28766	$ln requiere un argumento numérico positivo.
28812	Fieldpath está vacío
28818	Fieldpath no comienza con '$'
31022	Este operador requiere que se especifique un parámetro de «entrada».
31023	Este operador requiere un parámetro de «expresión regular».
31024	El operador recibe un argumento desconocido.
31032	Un nombre de campo no puede contener un byte nulo incrustado.
31034	Un valor está fuera del rango permitido de [-32768, 32767].
31095	WeekYear'iso' debe dar como resultado un número entero entre 1 y 9999.
31138	El valor $meta utilizado para la clasificación no es una palabra clave de metadatos válida.
31170	Se esperaba un objeto vacío, pero se proporcionó un valor no vacío.
31249	Una trayectoria de proyección choca con otro campo de la proyección.

Código de error	Description (Descripción)
31254	No se puede especificar una inclusión dentro de una proyección de exclusión.
31393	$bsonSize requiere un documento como entrada.
31394	Un operador de proyección posicional no puede aparecer en medio de una trayectoria de campo.
34413	Se usó un operador no válido en una consulta geográfica cercana.
34435	El argumento de $reverseArray debe ser una matriz.
34443	$range requiere un valor inicial numérico.
34444	$range requiere un valor inicial que se pueda representar como un entero de 32 bits.
34445	$range requiere que su valor final sea numérico.
34446	El valor final del rango debe caber en un entero de 32 bits.
34447	$range requiere un valor de paso numérico.
34448	$range requiere un valor de paso que se pueda representar como un entero de 32 bits.
34449	$range requiere un valor de paso distinto de cero
34450	El índice inicial de $substrCP debe ser de tipo numérico.
34451	El índice inicial $SubstrCP no se puede representar como un entero de 32 bits.
34452	El argumento de longitud de $substrCP no es de tipo numérico.
34453	La longitud $substrCP no se puede representar como un entero de 32 bits.
34454	$substrCP: la longitud debe ser un entero no negativo.
34455	$substrCP: el índice inicial debe ser un entero no negativo.

Código de error	Description (Descripción)
34460	$zip solo admite un objeto como argumento.
34461	Las entradas $zip deben ser una matriz de expresiones.
34462	La opción «predeterminada» debe ser una matriz de expresiones.
34463	la LongestLength opción de uso de $zip debe ser booleana.
34464	Se le dio a $zip un argumento desconocido.
34465	$zip requiere al menos una matriz de entrada
34466	no puede especificar los valores predeterminados a menos que el uso sea verdadero LongestLength
34467	los valores predeterminados y las entradas deben tener la misma longitud
34468	$zip recibe una expresión que no es matricial en su entrada.
34471	$strlEncp requiere un argumento de cadena.
34473	$str LenBytes requiere un argumento de cadena.
40060	$switch requiere un objeto como argumento.
40061	$switch requiere que las «ramas» sean una matriz.
40062	$switch requiere que cada rama sea un objeto.
40063	$switch recibe un argumento desconocido dentro de una rama.
40064	$switch requiere que cada rama tenga una expresión en «mayúsculas y minúsculas».
40065	$switch requiere que cada rama tenga una expresión «entonces».
40066	$switch no pudo encontrar una rama coincidente para una entrada y no se especificó ningún valor predeterminado.

Código de error	Description (Descripción)
40067	La expresión $switch contiene un argumento desconocido.
40068	$switch requiere al menos una sucursal
40075	$reduce requiere un objeto
40076	Argumento desconocido para $reduce
40077	$reduce requiere 'entrada'
40078	$reduce requiere 'initialValue'
40079	$reduce requiere «en»
40080	La entrada a $reduce debe ser una matriz.
40081	El operador requiere una matriz como segundo argumento.
40085	$split requiere su primer argumento para dar como resultado una cadena.
40086	El segundo argumento de $split debe dar como resultado una cadena.
40087	$split requiere un separador que no esté vacío
40090	$index OfArray requiere una matriz como primer argumento
40091	El primer argumento de $index no OfBytes es una cadena.
40092	$index OfBytes requiere una cadena como segundo argumento.
40093	$indexOfCP requiere una cadena como primer argumento.
40094	$indexOfCP requiere una cadena como segundo argumento.
40096	$index OfArray requiere valores de índice iniciales y finales integrales.
40097	$index OfArray requiere índices numéricos iniciales y finales no negativos.

Código de error	Description (Descripción)
40148	el ByCount campo de clasificación debe definirse como una ruta con el prefijo $ o una expresión dentro de un objeto
40149	el ByCount campo de clasificación debe especificarse como una cadena o como un objeto
40156	el campo de recuento debe ser una cadena que no esté vacía
40157	el campo de recuento debe ser una cadena que no esté vacía
40158	el campo de recuento no puede ser una ruta con el prefijo $
40159	el campo de recuento no puede contener un byte nulo
40160	el campo de recuento no puede contener '.'
40169	la especificación $facet debe ser un objeto no vacío
40170	los argumentos de $facet deben ser matrices
40171	los elementos de las matrices en la especificación $facet no deben ser objetos vacíos
40172	la subcanalización de la etapa $facet no puede estar vacía
40176	Se han dado nombres de campo contradictorios
40177	$project y $addFields requieren al menos un campo de salida.
40178	No se pueden excluir campos excepto '_id' en una proyección de inclusión
40179	No se pueden incluir campos ni restablecer campos ni agregar campos calculados en una proyección de exclusión
40180	$project no permite un objeto vacío como valor de campo.
40181	$project:: no válido debido a:: una especificación de expresión debe contener exactamente un campo.

Código de error	Description (Descripción)
40183	No se puede usar un nombre de campo punteado dentro de un subobjeto.
40192	El campo 'boundaries' del $bucket debe tener al menos 2 valores
40193	Todos los valores de la opción «boundaries» de $bucket deben ser del mismo tipo
40194	La opción «boundaries» de $bucket debe estar ordenada, pero los elementos no están en orden ascendente
40195	El campo 'default' de $bucket debe ser una expresión constante
40196	El campo 'output' del $bucket debe ser un objeto
40197	Opción no reconocida para $bucket
40198	$bucket requiere que se especifiquen 'groupBy' y 'boundaries'
40199	El campo 'predeterminado' del $bucket debe ser menor que el límite inferior o mayor o igual que el límite más alto.
40200	El campo 'boundaries' de $bucket debe ser una matriz
40201	El argumento de $bucket stage debe ser un objeto
40202	El campo 'groupBy' del $bucket debe definirse como una ruta o expresión con el prefijo $
40203	El campo debe ser un objeto acumulador
40204	El nombre del campo no puede contener '.'
40205	El nombre del campo no puede ser el nombre de un operador
40206	El acumulador es un operador unario
40218	la consulta requiere metadatos de puntuación de texto, pero no está disponible

Código de error	Description (Descripción)
40228	La expresión 'newRoot' de $replaceRoot debe evaluarse como un objeto.
40229	$replaceRoot requiere un objeto como especificación.
40230	$replaceRoot solo acepta la opción 'newRoot'.
40234	El campo $group debe ser un objeto acumulador.
40235	Un nombre de campo no puede contener el carácter «.»
40236	Los nombres de campo no pueden empezar por '$'
40237	Este acumulador es un operador unario y acepta un solo argumento.
40239	El campo 'GroupBy' de $bucketAuto debe definirse como una ruta con el prefijo $ o un objeto de expresión
40240	El argumento de $bucketAuto debe ser un objeto
40241	El campo 'buckets' de $bucketAuto debe ser un valor numérico
40242	El campo 'buckets' de $bucketAuto debe poder representarse como un entero de 32 bits
40243	El campo 'buckets' de $bucketAuto debe ser mayor que 0
40244	El campo de «salida» de $bucketAuto debe ser un objeto
40245	Opción no reconocida para $bucketAuto
40246	$bucketAuto requiere que se especifiquen 'groupBy' y 'buckets'
40257	Granularidad de redondeo desconocida
40258	$bucketAuto puede especificar una «granularidad» solo con límites numéricos
40260	$bucketAuto puede especificar una «granularidad» solo con números no negativos, pero encontró un número negativo

Código de error	Description (Descripción)
40261	El campo 'granularity' de $bucketAuto debe ser una cadena
40272	La especificación de etapa $addFields debe ser un objeto.
40319	la especificación de la etapa $lookup debe ser un objeto
40320	falta la opción 'from' en la especificación de la etapa $lookup
40321	La opción 'from' para $lookup debe ser una cadena
40322	espacio de nombres $lookup no válido
40323	Un objeto de especificación de una fase de canalización debe contener exactamente un campo.
40324	La canalización contiene un nombre artístico no reconocido.
40331	Esta etapa de agregación no está permitida dentro de una canalización de $facet.
40352	Las rutas de los campos de $project no pueden ser cadenas vacías.
40353	La ruta de un campo no debe terminar en «.».
40386	$array ToObject requiere una entrada de matriz.
40390	$object ToArray requiere un documento como entrada.
40391	$array ToObject requiere un formato de entrada coherente. Todos los elementos deben ser matrices o todos deben ser objetos. Se detectó el objeto, ahora se ha encontrado: matriz
40392	$array ToObject requiere objetos con exactamente las claves 'k' y 'v'.
40393	$array ToObject requiere un objeto con las claves 'k' y 'v'. Faltan una o ambas claves
40394	$array ToObject requiere objetos con claves de cadena «k» y valores «v».

Código de error	Description (Descripción)
40395	$array ToObject requiere pares clave-valor cuyas claves sean cadenas.
40396	$array ToObject requiere un formato de entrada coherente. Todos los elementos deben ser matrices o todos deben ser objetos. Se detectó la matriz, ahora se ha encontrado: objeto
40397	$array ToObject requiere una matriz de matrices de dos elementos.
40398	$array ToObject recibe una entrada de un tipo o formato no reconocido.
40400	$mergeObjects requiere que todas las entradas sean objetos.
40414	Falta un campo de comando obligatorio o no se pudo validar.
40415	El comando contiene un campo no reconocido.
40422	El nombre de un campo de matriz BSON tiene un valor no válido.
40485	Se proporcionó un identificador de zona horaria no reconocido.
40489	$date FromParts no permite mezclar campos de calendario normales con campos de fecha de semana ISO
40515	Un FromParts argumento $date no se evalúa como un número entero.
40516	$date FromParts requiere que estén presentes «year» o «isoWeekYear»
40517	La opción «zona horaria» debe evaluarse como una cadena para los operadores de fecha.
40518	$date FromParts o $date ToParts obtuvieron un parámetro no reconocido o mal escrito.
40519	$date FromParts solo admite un objeto como argumento.
40520	ToParts Se le asignó un parámetro no reconocido a $date.
40522	$date ToParts requiere un argumento de objeto con un parámetro «fecha».

Código de error	Description (Descripción)
40523	$date FromParts 'year' debe ser un número entero comprendido entre 1 y 9999.
40525	week/day Los valores FromParts ISO de $date están fuera de rango o faltan campos obligatorios.
40533	El argumento de la zona horaria debe ser una cadena.
40535	Se proporcionó una opción no reconocida al escenario o a la expresión.
40536	La expresión acepta exactamente un argumento cuando se le da una matriz.
40539	Falta el argumento de «fecha» obligatorio para un operador de fecha.
40540	$date FromString solo admite un objeto como argumento
40541	$date FromString recibió un argumento no reconocido.
40542	Falta el parámetro 'dateString' para $date FromString
40543	$date FromString requiere que 'dateString' sea una cadena.
40545	La date/time cadena está incompleta y faltan los elementos necesarios.
40551	no puede pasar una date/time cadena con información de zona horaria junto con un argumento de zona horaria
40553	$date FromString no puede analizar la cadena de fecha dada.
40554	no puedes pasar una date/time cadena con un desfase GMT junto con un argumento de zona horaria
40575	$ChangeStream 'fullDocument' debe ser «predeterminado» o «UpdateLookup».
40600	$out no se puede usar en una etapa de $facet.
40601	$out y $merge solo pueden ser la etapa final del proceso.
40602	Una fase como $changeStream o $geoNear no es la primera en preparación.

Código de error	Description (Descripción)
40603	$GeoNear no era la primera etapa en preparación después de la optimización. ¿La optimización está deshabilitada o inhibida?
40647	Al token de reanudación del flujo de cambios le falta un campo _data o está mal escrito.
40674	Solo se permite un tipo de opción de currículum, pero se encontraron varias.
50001	la opción automática no IndexId:false está permitida para la colección especific ada.
50694	A este operador se le dio un argumento desconocido.
50695	Este operador requiere un campo de «entrada».
50696	El operador solo admite un objeto como argumento.
50699	Este operador de expresión requiere que su entrada sea una cadena.
50700	El argumento 'chars' debe ser una cadena.
50736	GetMore no se puede ejecutar en una sesión con un cursor no creado en una sesión.
50737	Para ejecutar GetMore en un cursor creado por una sesión es necesario proporcionar un lsid.
50752	El índice inicial de $SubstrBytes es negativo.
50808	$changeStream stage espera un documento como argumento.
51024	El valor de «fuerza» de la intercalación debe ser de 5 o menos.
51047	Este operador o etapa no está permitido dentro de una subcanalización de $lookup.
51075	No se pueden establecer opciones tanto en $regex como en $options

Código de error	Description (Descripción)
51076	Memoria insuficiente para crear el índice vectorial para el número especificado de listas.
51078	No se pudo encontrar el nombre del índice.
51079	No se admiten las estadísticas de latencia
51091	Una expresión regular no es válida o está mal formada.
51103	El operador espera un objeto de argumentos con nombre, pero recibió otro tipo.
51104	El operador requiere que su argumento de «entrada» sea una cadena.
51105	El argumento 'regex' de un operador no es una cadena o un tipo de expresión regular.
51106	Un operador requiere que su parámetro «opciones» sea una cadena.
51107	Las opciones de $regex no se pueden especificar en los campos «expresiones regulares» y «opciones».
51108	Las opciones de expresiones regulares contienen un indicador no válido o un byte nulo incrustado.
51109	Una expresión regular no puede contener un byte nulo incrustado.
51110	Las opciones de expresiones regulares no pueden contener un byte nulo incrustado.
51111	La expresión regular proporcionada a este operador no es válida.
51151	El tamaño del búfer para almacenar la salida superó el límite de tipos de datos
51156	No se pudo ejecutar una expresión regular en $regex durante la evaluación.
51270	Un objeto de subproyección vacío no es un valor de proyección válido.
51272	Error de sintaxis: $projection requiere al menos un campo de salida.

Código de error	Description (Descripción)
51744	El operador requiere que la palabra «reemplazo» sea una cadena.
51745	El operador requiere que su argumento de «búsqueda» sea una cadena.
51746	El argumento de «entrada» de un operador no es una cadena.
51747	Un operador necesita un parámetro de «reemplazo».
51748	Este operador requiere un parámetro de «búsqueda».
51749	Este operador requiere un parámetro de «entrada».
51750	Un operador de agregación recibió un argumento desconocido.
100003	Error interno del sistema al procesar la ruta de consulta optimizada
100005	Se ha superado un límite configurado, como recopilaciones, funciones personali zadas o privilegios.
100006	Se ha superado el número máximo de usuarios permitidos.
3040501	$rand no acepta argumentos actualmente
3041701	$getField recibió un argumento desconocido en $project.
3041702	$project:: inválido debido a:: $getField requiere que se especifique un 'campo'
3041703	$project:: inválido debido a:: $getField requiere que se especifique una 'entrada'
3041704	$getField requiere el argumento 'field' para dar como resultado una cadena.
4161100	$setField y $unsetField solo admiten un objeto como argumento.
4161101	A $setField o $unsetField se le asignó un argumento desconocido.
4161102	$setField y $unsetField requieren que se especifique un «campo».
4161103	$project:: inválido causado por:: $setField requiere que se especifique un 'valor'

Código de error	Description (Descripción)
416105	$setField o $unsetField requieren una «entrada» para evaluar un objeto.
4161106	$setField y $unsetField requieren 'field' para obtener una constante.
4161107	$setField o $unsetField requieren 'field' para dar como resultado una cadena.
4161108	$project no permite hacer referencia a una ruta de campo aquí; usa {$literal:...} en su lugar.
4161109	$setField y $unsetField requieren que se especifique un campo de «entrada».
4662500	El comando contiene un campo no reconocido.
4862100	El espacio de nombres completo supera la longitud máxima permitida.
5166301	$DateDiff solo admite un objeto como argumento
5166302	$DateDiff recibió un argumento no reconocido.
5166303	Falta el parámetro 'startDate' en $DateDiff
5166304	Falta el parámetro 'endDate' en $DateDiff
5166305	Falta el parámetro «unidad» en $DateDiff
5166307	Un argumento $dateDiff no es una fecha.
5166308	$DateDiff se desbordó al calcular la diferencia de fechas.
5166400	$dateAdd y $dateSubtract requieren que su argumento sea un objeto.
5166401	$dateAdd o $dateSubtract tienen un argumento no reconocido.
5166402	$dateAdd o $dateSubtract requieren que estén presentes la fecha de inicio, la unidad y el importe.
5166403	$dateAdd requiere que StartDate se pueda convertir en una fecha
5166404	$dateAdd espera una cadena que defina la unidad de tiempo

Código de error	Description (Descripción)
5166405	$DateAdd espera una cantidad entera de unidades de tiempo
5166406	DateAdd desbordado
5166503	Se ha proporcionado un valor de unidad de tiempo desconocido.
5439002	$DateTrunc se desbordó al calcular la fecha truncada.
5439006	El valor BinSize de DateTrunc no es compatible
5439007	$dateTrunc solo admite un objeto como argumento
5439008	Argumento $DateTrunc no reconocido; fecha esperada, unidad, binSize, zona horaria, inicio. OfWeek
5439009	Falta el parámetro 'date' para $dateTrunc
5439010	Falta el parámetro «unidad» en $dateTrunc
5439012	$dateTrunc requiere que «fecha» sea una fecha.
5439013	$dateDiff y $dateTrunc requieren que 'unit' sea una cadena.
5439014	El valor de «unidad» de $DateDiff y $DateTrunc no es una unidad de tiempo reconocida.
5439015	$dateDiff o $dateTrunc requieren que «start» sea una cadena. OfWeek
5439016	El valor de «inicioOfWeek» de $dateDiff o $dateTrunc no es un día de la semana válido.
5439017	$dateTrunc requiere que 'binSize' sea un entero de 64 bits.
5439018	$dateTrunc requiere que 'binSize' sea mayor que 0.
5447000	La etapa $collStats debe ser un objeto anidado.

Código de error	Description (Descripción)
5626500	$GeoNear, $near y $nearSphere no están permitidos en este contexto, ya que estos operadores requieren ordenar los datos geoespaciales. Si no necesita ordenarlos, considere usar $GeoWithin en su lugar.
5654600	$project no permite hacer referencia a una ruta de campo en este contexto.
5654601	$project:: no válido debido a que:: $getField requiere 'field' para evaluarse como una constante, pero tiene un argumento no constante
5654602	$getField requiere un «campo» para dar como resultado una cadena.
5733415	no se puede modificar $mod by cero
5860400	$geoNear requiere un argumento «cercano»
5887502	Todos los operandos de $setEquals deben ser matrices.
5911200	PlanExecutor error durante la agregación:: causado por:: $mergeObjects solo admite objetos
5976500	Se ha asignado a $DateAdd un valor de parámetro «importe» no válido.
6045000	Se ha asignado a $DateSubtract un valor de parámetro «importe» no válido.
7158303	$switch no encontró ninguna rama coincidente para la entrada y no se especificó ningún valor predeterminado.

Notas de la versión

Estas notas de la versión describen las funciones, mejoras y correcciones de errores de Amazon DocumentDB organizadas por versión principal del motor. En cada sección de versión, las entradas se enumeran por fecha en formato de tabla.

Para obtener más información sobre cómo se numeran las versiones del motor y las versiones de los parches (incluida la asignación entre los prefijos de las versiones de los parches del motor y las versiones del motor de Amazon DocumentDB), consulte. Numeración de versiones del motor

Para determinar la versión de parche actual del motor de Amazon DocumentDB, ejecute el siguiente comando:

```
db.runCommand({getEngineVersion: 1})
```

> **Note**
>
> A partir de Amazon DocumentDB 5.0.2 y 8.0.2, este comando devuelve la versión del motor y la versión del parche del motor como una sola cadena en el formulario. *`engine-version`*`+`*`engine-patch-version`* Por ejemplo, un clúster de Amazon DocumentDB 5.0.2 devuelve y un clúster de `5.0.2+3.0.12345` 8.0.2 devuelve. `8.0.2+4.0.12345` Los clústeres de versiones anteriores devuelven la versión del parche del motor por sí solos, por ejemplo. `3.0.12345` Para obtener más información, consulte Numeración de versiones del motor.

Si su clúster no está en la versión más reciente del motor, es probable que tenga disponibles tareas de mantenimiento pendientes que permitan actualizar el motor. Para obtener más información, consulte Mantenimiento de Amazon DocumentDB en la Guía para desarrolladores.

Temas

- Notas de lanzamiento de la versión del motor Amazon DocumentDB
- Notas generales de la versión de Amazon DocumentDB
- Notas de la versión de Amazon DocumentDB Elastic Clusters

Notas de lanzamiento de la versión del motor Amazon DocumentDB

Notas de versión de las versiones del motor Amazon DocumentDB organizadas por versión principal.

Amazon DocumentDB 8.0

Notas de versión de la versión 8.0 del motor Amazon DocumentDB (compatibilidad con MongoDB 8.0).

Amazon DocumentDB 8.0.2

Date	Notas de la versión
28 de septiembre de 2026	Amazon DocumentDB 8.0.2 (versión del motor: 4.0.12042) Nuevas características y mejoras 1. Se agregó soporte para la etapa de `$setWindowFields` agregación y sus operadores de ventanas en la versión 3 de Planner. 2. Se agregó soporte para la etapa de `$bucketAuto` agregación en la versión 3 de Planner. 3. Se agregó soporte para la etapa de `$facet` agregación en la versión 3 de Planner. 4. Se agregó soporte para la etapa de `$graphLookup` agregación en la versión 3 de Planner. 5. En la versión 3 de Planner, se agregó soporte para `$lookup` subconsultas correlacionadas, incluida la sintaxis concisa de subconsultas correlacionadas, con un subconjunto de etapas de canalización. 6. Se ha añadido soporte para la etapa de agregación. `$changeStreamSplitLargeEvent` 7. Los flujos de cambios ahora emiten eventos para `createCollection` las operaciones.

Date	Notas de la versión
	8. Los flujos de cambios ahora emiten eventos para `createIndex` las operaciones. 9. Se ha mejorado el rendimiento de escritura de los flujos de cambios. 10. Se agregó soporte para escrituras que se pueden volver a intentar. 11. Se agregó soporte para el `reIndex` comando en índices parciales. 12. La versión 3 de Planner ahora admite el escaneo de índices en `$expr` predicados. 13. La versión 3 de Planner introduce escaneos de campos múltiples únicamente basados en índices (IXONLYSCAN) para realizar consultas de búsqueda. Para obtener más información, consulte Index-only escaneos (consultas cubiertas). 14. La versión 3 de Planner introduce escaneos de solo índice (IXONLYSCAN) para etapas de agregación como. `$group` Para obtener más información, consulte Index-only escaneos (consultas cubiertas). 15. Optimización del rendimiento para `$count` y `countDocuments()` en la versión 3 de Planner. 16. Se agregó soporte para la clasificación incremental en la versión 3 de Planner. Para obtener más información, consulte Ordenación incremental. 17. Se ha mejorado la selección de planes para las consultas de índice de texto. 18. El plan de explicación para las `$unwind` etapas combinadas `$lookup` ahora refleja el `includeArrayIndex` parámetro. 19. Se agregó compatibilidad con filtros de índice en todos los comandos. 20. El planificador de consultas ahora prioriza los índices únicos para los predicados de equality ()`$eq`. 21. Se ha aumentado el límite de truncamiento de los comandos agregados en la salida. `currentOp`

Date	Notas de la versión
	Correcciones de errores 1. Se ha corregido un error en la versión 1 (predeterminada) del planificador de consultas. 2. Se ha corregido el tratamiento de ida y vuelta del texto de expresiones de índice parciales durante la actualización de la versión principal (MVU). 3. Se ha corregido un problema con el generador de perfiles para las consultas de recuento rápido. 4. Solución de seguridad para el `collStats` comando.

Amazon DocumentDB 8.0.1

Date	Notas de la versión
15 de septiembre de 2026	Mejoras de seguridad y estabilidad Amazon DocumentDB 8.0.1 (versión del motor: 4.0.11638) [en curso] Amazon DocumentDB aplica automáticamente este parche durante el siguiente período de mantenimiento.
31 de agosto de 2026	Nueva característica Amazon DocumentDB ahora admite actualizaciones de versiones principales (MVU) in situ directamente desde las versiones 3.6 y 4.0 del motor a la versión 8.0. Esto elimina la necesidad de actualizar las versiones intermedias y le permite actualizar los clústeres 3.6 o 4.0 a la vez que conserva los datos, las configuraciones y los ajustes del clúster existentes. Las principales actualizaciones de las versiones 3.6 y 4.0 de Amazon DocumentDB a la 8.0 están disponibles en todas AWS las regiones en las que actualmente se admiten estas versiones . Para obtener más información, consulte Actualización local de la versión principal Amazon DocumentDB.

Date	Notas de la versión
18 de agosto de 2026	Mejoras de seguridad y estabilidad Amazon DocumentDB 8.0.1 (versión del motor: 4.0.9938) Este parche del motor proporciona las siguientes mejoras: • Procesamiento de solicitudes mejorado: Amazon DocumentDB ahora valida las solicitudes entrantes de los clientes de forma más exhaustiva a lo largo de la ruta de conexión y de análisis de mensajes. Esto mejora la resiliencia y la estabilidad en una gama más amplia de patrones de entrada. • Amazon DocumentDB, que refuerza la autenticación, ahora valida el intercambio de autenticación de forma más exhaustiva. Esto mejora la solidez a la hora de gestionar entradas inesperadas o con un formato incorrecto. Amazon DocumentDB aplica automáticamente este parche durante el siguiente período de mantenimiento.
11 de agosto de 2026	Nuevas características Amazon DocumentDB 8.0.1 (versión del motor: 4.0.8864) • Se ha habilitado el soporte para la autenticación. SCRAM-SHA-256 • Se habilitó el RPO ajustable para los clústeres globales de Amazon DocumentDB. • Se mejoró la recuperación del almacenamiento durante la recolección de basura en segundo plano para reducir el espacio de almacenamiento no utilizado con el tiempo.
22 de julio de 2026	Nueva característica Los clústeres de Amazon DocumentDB que ejecutan la versión 8.0 y posteriores del motor ahora admiten un límite máximo de almacenamiento de 256 TiB, lo que duplica el límite anterior de 128 TiB. Para obtener más información, consulte Cuotas de Amazon DocumentDB.

Date	Notas de la versión
13 de julio de 2026	Amazon DocumentDB 8.0.1 (versión del motor: 4.0.8864) Nuevas características y mejoras 1. Soporte para 13 operadores de acumulador en el planificador predeterminado (v3):`$top`,,,,,,,`$bottom`,,`$topN`,`$bottomN`,`$count`,`$firstN`. `$lastN` `$minN` `$maxN` `$median` `$percentile` `$stdDevPop` `$stdDevSamp` 2. Soporte para 36 operadores de expresión en Planner predeterminado (v3):`$median`,`$percentile` ,`$stdDevPop` ,`$stdDevSamp` ,`$round`,`$sigmoid`,,`$trunc`,`$firstN`,`$lastN`,`$maxN`,`$minN`,`$bitAnd`,`$bitOr`,`$bitXor`,`$bitNot`,`$bitAnd`,`$bitOr`,`$bitXor`,`$bitNot`,`$size` ,`$bsonSize` ,`$tsIncrement` ,`$tsSecond` ,`$cos`,`$acos`,,`$cosh`,`$acosh`,`$sin`,`$asin`,`$sinh`,`$asinh`,`$tan`,,`$atan`,`$atan2`$atanh`,`$degreesToRadians` . `$radiansToDegrees` `$isNumber` `$toUUID` 3. Soporte para `$sampleRate` Planner predeterminado (v3). `$match` 4. Soporte para `$sortByCount` Planner predeterminado (v3). 5. Se agregó soporte para la etapa de `$listSearchIndexes` agregación en el Planner predeterminado (v3). Devuelve los metadatos del índice de búsqueda (nombre, estado, consultable, versión) para los índices de búsqueda de texto y vectores de una colección. 6. Admite una compresión rápida mediante el tipo de mensaje OP_COMPRESSED. 7. Admite los métodos `min` y `max` cursor del comando de búsqueda en el Planner predeterminado (v3). 8. La versión 3 de Planner introduce escaneos de solo índice (IXONLYSCAN) para. `countDocuments()` 9. Se ha añadido la compatibilidad con índices ocultos para todos los tipos de índices (versión 2 y superiores del planificador).

Date	Notas de la versión
	Correcciones de errores 1. Se ha corregido un problema por el que las consultas de `$text` búsqueda que contienen caracteres de subrayado podían arrojar resultados incorrectos si se evaluaban como un filtro residual.

Amazon DocumentDB 8.0.0

Date	Notas de la versión
15 de septiembre de 2026	Mejoras de seguridad y estabilidad Amazon DocumentDB 8.0.0 (versión del motor: 4.0.11440) [en curso] Correcciones de errores Esta versión incluye la siguiente corrección de errores: 1. Se ha corregido un problema por el que las consultas de `$text` búsqueda que contienen caracteres de subrayado podían arrojar resultados incorrectos si se evaluaban como un filtro residual. Amazon DocumentDB aplica automáticamente este parche durante el siguiente período de mantenimiento.
18 de agosto de 2026	Mejoras de seguridad y estabilidad Amazon DocumentDB 8.0.0 (versión del motor: 4.0.9912) Este parche del motor proporciona las siguientes mejoras: • Procesamiento de solicitudes mejorado: Amazon DocumentDB ahora valida las solicitudes entrantes de los clientes de forma más exhaustiva a lo largo de la ruta de conexión y de análisis de mensajes. Esto mejora la resiliencia y la estabilidad en una gama más amplia de patrones de entrada.

Date	Notas de la versión
	• Amazon DocumentDB, que refuerza la autenticación, ahora valida el intercambio de autenticación de forma más exhaustiva. Esto mejora la solidez a la hora de gestionar entradas inesperadas o con un formato incorrecto. Si no lo aplica antes, Amazon DocumentDB aplicará automáticamente este parche durante el siguiente período de mantenimiento, 30 días después del lanzamiento.
22 de julio de 2026	Nueva característica Los clústeres de Amazon DocumentDB que ejecutan la versión 8.0 y posteriores del motor ahora admiten un límite máximo de almacenam iento de 256 TiB, lo que duplica el límite anterior de 128 TiB. Para obtener más información, consulte Cuotas de Amazon DocumentDB.
13 de mayo de 2026	Nueva característica Amazon DocumentDB serverless ya está disponible para Amazon DocumentDB 8.0. Para obtener más información, consulte Uso de Amazon DocumentDB sin servidor.
6 de mayo de 2026	Correcciones de errores y otros cambios Amazon DocumentDB 8.0 (versión del motor: 4.0.6940) Esta actualización proporciona un parche de motor opcional con las siguientes mejoras: 1. Se ha corregido un error en la recolección de basura que, en un caso extremo, podía provocar que los lectores se bloquearan.

Date	Notas de la versión
20 de abril de 2026	Nueva característica In-place Ya está disponible la actualización de la versión principal (MVU) de Amazon DocumentDB 5.0 a Amazon DocumentDB 8.0. Para obtener más información, consulte Actualización local de la versión principal Amazon DocumentDB.

Date	Notas de la versión
20 de abril de 2026	Nuevas características y mejoras Amazon DocumentDB 8.0 (versiones del motor: 4.0.5995, 4.0.6473) Se trata de un parche de motor obligatorio que incluye las siguientes funciones, mejoras y correcciones nuevas: 1. `$sortArray` operador en`$project`: • Soporte para el `$sortArray` operador en `$project` fase de agregación para la versión 3 de Planner. El `$sortArra` y operador ordena una matriz en función de sus elementos (para valores escalares) o según un campo específico (para matrices de documentos), y devuelve la matriz ordenada en orden ascendente (1) o descendente (). `-1` 2. Soporte para`$out`,,, con el uso de `$collStats` Planner `$indexStats` v3`$geoNear`. Con Planner V3, el `$out` escenario también permite escribir en colecciones de diferentes bases de datos. 3. La canalización de agregación ahora se fusiona `$lookup` y se `$unwind` escalona incluso cuando la `$unwind` etapa incluye el parámetro. `includeArrayIndex` Anteriormente, esta optimizac ión se omitía cuando `includeArrayIndex` estaba presente. 4. Mejoras en el rendimiento para mejorar el rendimiento de escritura de las instancias db.r8g.8xlarge y de mayor tamaño que se ejecuten según la configuración. I/O-Optimized 5. Ahora puede supervisar la métrica de las instancias. `Connectio ns (active)` Para obtener más información, consulta Cuotas de instancias. 6. Soporte para operadores de `$dateFromParts` agregación `$dateToParts` y soporte. 7. Command-level Las métricas de rendimiento ahora están disponibles en CloudWatch las operaciones de CRUD (`find`,,`insert`,`findAndModify` ,,`update`,`delete`,)`aggregate` ,`count`,`distinct`,`getMore`,`abortTran`

Date	Notas de la versión
	`saction` ,,,,`commitTransaction`) `AvgDuration` `P100Duration` , y en `MaxConcurrent` las dimensiones para mejorar la supervisión operativa. 8. Se ha corregido un error de `$replaceRoot` comportamiento por el que faltaban en el documento de salida `newRoot` campos que contenían valores booleanos de `true`0/`false`1o valores numéricos de/. 9. Se ha corregido un error en el `$indexOfArray` que las consultas válidas devolvían errores que indicaban que no eran válidas. 10. El plan de explicación ahora incluye `docsExamined` métricas en los `executionStats` resultados. 11. Rendimiento mejorado de la autenticación SASL. 12. Se optimizaron las actualizaciones de los documentos para reducir la sobrecarga de almacenamiento cuando los valores del índice permanecen inalterados y los documentos caben en las páginas existentes. 13. Soporte para `bypassDocumentValidation` in`$merge`. Todos los clústeres de Amazon DocumentDB 8.0 se actualizarán automáticamente a esta versión de parche durante su próximo período de mantenimiento.
26 de marzo de 2026	Nuevas características Amazon DocumentDB ahora admite la última generación de instancias optimizadas para memoria, db.r8g, con procesadores Arm-based AWS Graviton4, lo que proporciona un rendimiento hasta un 30% superior al de las instancias R6G. Para obtener Administración de clases de instancias más información, consulte esta guía

Date	Notas de la versión
11 de noviembre de 2025	Nuevas características Amazon DocumentDB ahora ofrece compatibilidad total con los protocolos de cable con MongoDB 8.0. Amazon DocumentDB 8.0 mejora el rendimiento de las consultas, lo que le permite crear aplicaciones de alto rendimiento a un costo menor. Amazon DocumentDB 8.0 (versión 4.0.4448 del parche del motor) • Se agregó compatibilidad con los controladores de API de MongoDB 8.0. • Se agregó compatibilidad con la versión 3 de Planner, que amplía las mejoras de rendimiento a los operadores de la fase de agregación, además de admitir la optimización de las canalizac iones de agregación y los distintos comandos. • Se agregó la compatibilidad con la compresión basada en diccionar ios mediante el algoritmo de compresión Zstandard, lo que mejora la relación de compresión 5 veces. • Nuevas capacidades añadidas: Amazon DocumentDB 8.0 admite la intercalación y las vistas. • Se agregó soporte para nuevas etapas y operadores de agregació n: $replaceWith, $VectorSearch, $merge, $set, $unset, $bucket y 3 nuevos operadores de agregación: $pow, $rand y $dateTrunc. • Se ha añadido compatibilidad con una nueva versión del índice de texto: el índice de texto v2 de Amazon DocumentDB 8.0 incorpora tokens adicionales que mejoran las capacidades de búsqueda de texto. • Incluye mejoras en la búsqueda vectorial mediante la creación de índices vectoriales en paralelo, lo que reduce el tiempo de creación de índices hasta 30 veces. • Se ha eliminado la compatibilidad con TLS 1.0 y TLS 1.1. La versión mínima de TLS ahora es TLS 1.2.

Amazon DocumentDB 5.0

Notas de versión de la versión 5.0 del motor Amazon DocumentDB (compatibilidad con MongoDB 5.0).

Amazon DocumentDB 5.0.2

Date	Notas de la versión
28 de septiembre de 2026	Amazon DocumentDB 5.0.2 (versión del motor: 3.0.23065) Nuevas características y mejoras 1. Se agregó la compatibilidad con índices ocultos para todos los tipos de índices (versión 2 y posteriores del planificador). 2. Soporte para los métodos `min()` y `max()` cursor en el comando de búsqueda. 3. Se agregó compatibilidad con filtros de índice en todos los comandos. 4. El planificador de consultas ahora prioriza los índices únicos para los predicados de equality ()`$eq`. 5. Se ha aumentado el límite de truncamiento de los comandos agregados en la salida. `currentOp` Correcciones de errores 1. Se ha corregido un error en la versión 1 (predeterminada) del planificador de consultas. 2. Se ha corregido el tratamiento de ida y vuelta del texto de expresiones de índice parciales durante la actualización de la versión principal (MVU). 3. Se ha corregido un problema con el generador de perfiles para las consultas de recuento rápido.

Amazon DocumentDB 5.0.1

Date	Notas de la versión
18 de agosto de 2026	Mejoras de seguridad y estabilidad Amazon DocumentDB 5.0.1 (versión del motor: 3.0.22319) [en curso] Este parche del motor proporciona las siguientes mejoras: • Procesamiento de solicitudes mejorado: Amazon DocumentDB ahora valida las solicitudes entrantes de los clientes de forma más exhaustiva a lo largo de la ruta de conexión y de análisis de mensajes. Esto mejora la resiliencia y la estabilidad en una gama más amplia de patrones de entrada. • Amazon DocumentDB, que refuerza la autenticación, ahora valida el intercambio de autenticación de forma más exhaustiva. Esto mejora la solidez a la hora de gestionar entradas inesperadas o con un formato incorrecto. Si no lo aplica antes, Amazon DocumentDB aplicará automáticamente este parche durante el siguiente período de mantenimiento, 30 días después del lanzamiento.
11 de agosto de 2026	Nueva característica Amazon DocumentDB 5.0.1 (versión del motor: 3.0.21138) Se ha habilitado el soporte para la autenticación. SCRAM-SHA-256
8 de junio de 2026	Nueva versión secundaria Amazon DocumentDB 5.0.1 (versión del motor 3.0.21138) Amazon DocumentDB 5.0.1 es la primera versión secundaria de la versión principal de Amazon DocumentDB 5.0. Las versiones secundarias incluyen mejoras como nuevas funciones, correcciones de seguridad y correcciones de errores. Puedes actualizar a la 5.0.1 desde la 5.0.0 mediante una actualización de versión secundaria.

Date	Notas de la versión
	Para obtener más información, consulte Actualización de la versión secundaria de Amazon DocumentDB. Esta versión secundaria incluye las siguientes funciones y mejoras nuevas: 1. Se agregaron `gcRuntimeStats` y dividieron las métricas del nivel de recolección `storageSegmentExtended` en `storageSegmentBase` y dentro `collstats` para obtener más información sobre el estado general del proceso de recolección de basura. `documentFragmentStats` Para obtener más información, consulte Recopilación de elementos no utilizados en Amazon DocumentDB. 2. Soporte para operadores de `$pow` agregación `$rand` y de agregación. 3. Ahora puede supervisar la `Connections (active)` métrica en busca de instancias. Para obtener más información, consulta Cuotas de instancias. 4. Se mejoró la resiliencia interregional de Global Cluster al eliminar el reinicio secundario del clúster con el modo de solo lectura de Global Survivable. 5. Soporte para operadores de `$dateFromParts` agregación `$dateToParts` y soporte. 6. Command-level Las métricas de rendimiento ahora están disponibles en CloudWatch las operaciones de CRUD (`find`,,`insert`,`findAndModify` ,,`update`,`delete`,)`aggregate` ,`count`,`distinct`,`getMore`,`abortTran saction` ,,,,`commitTransaction`) `AvgDuration` `P100Durat ion` , y en `MaxConcurrent` las dimensiones para mejorar la supervisión operativa. 7. `$replaceRoot` genera `newRoot` campos que contienen valores booleanos (`true`/`false`) y valores numéricos (/). `0` `1` 8. `$indexOfArray` admite una gama más amplia de expresiones de consulta válidas.

Date	Notas de la versión
	9. El plan de explicación incluye `docsExamined` métricas en los `executionStats` resultados. 10. Rendimiento mejorado de la autenticación SASL.

Amazon DocumentDB 5.0.0

Date	Notas de la versión
18 de agosto de 2026	Mejoras de seguridad y estabilidad Amazon DocumentDB 5.0.0 (versión del motor: 3.0.22303) [en curso] Este parche del motor proporciona las siguientes mejoras: • Procesamiento de solicitudes mejorado: Amazon DocumentDB ahora valida las solicitudes entrantes de los clientes de forma más exhaustiva a lo largo de la ruta de conexión y de análisis de mensajes. Esto mejora la resiliencia y la estabilidad en una gama más amplia de patrones de entrada. • Amazon DocumentDB, que refuerza la autenticación, ahora valida el intercambio de autenticación de forma más exhaustiva. Esto mejora la solidez a la hora de gestionar entradas inesperadas o con un formato incorrecto. • Access-control cumplimiento Amazon DocumentDB ahora verifica de manera uniforme los privilegios requeridos en todos los niveles de anidamiento para las operaciones de consulta anidadas. Si no lo aplica antes, Amazon DocumentDB aplicará automáticamente este parche durante su próximo período de mantenimiento, 30 días después del lanzamiento.
6 de mayo de 2026	Correcciones de errores y otros cambios Amazon DocumentDB 5.0 (versión del motor: 3.0.20421)

Date	Notas de la versión
	Esta versión es para un parche de seguridad obligatorio.
20 de abril de 2026	Nueva característica In-place Ya está disponible la actualización de la versión principal (MVU) de Amazon DocumentDB 5.0 a Amazon DocumentDB 8.0. Para obtener más información, consulte Actualización local de la versión principal Amazon DocumentDB.
13 de abril de 2026	Correcciones de errores y otros cambios Amazon DocumentDB 5.0 (versiones del motor: 3.0.19917) Se trata de un parche de motor obligatorio que incluye las siguientes funciones, mejoras y correcciones nuevas: 1. La `DatabaseConnectionsActiveMax` métrica ahora hace un seguimiento `Connections (active)` de las instancias. Para obtener más información, consulta Cuotas de instancias. 2. Rendimiento mejorado de la autenticación SASL. Todos los clústeres de Amazon DocumentDB 5.0 se actualizarán automáticamente a esta versión de parche durante el próximo período de mantenimiento.

Date	Notas de la versión
22 de octubre de 2025	Nuevas características Amazon DocumentDB 5.0 (versión 3.0.17983 del parche de motor) Amazon DocumentDB ahora admite la última generación de instancia s optimizadas para memoria, db.r8g, con procesadores Arm-based AWS Graviton4, que proporcionan un rendimiento hasta un 30% superior al de las instancias R6G. Para obtener más información, consulte Administración de clases de instancias en esta guía y este artículo del blog. Nuevas características Amazon DocumentDB 5.0 (versión 3.0.15902 del parche del motor) • Se agregó soporte para nombres de colección más largos (longitud del espacio de nombres de hasta 255 caracteres). • Se introdujo un nuevo planificador de consultas (versión 2.0). Para obtener más información, consulte la sección Planificador de consultas v2 de esta guía. • Se agregaron nuevos campos `collstats` para «mvccidStats» y «gcRuntimeStats» que proporcionan información sobre el estado general de la recolección en relación con la recolección de basura y la recolección anticuada.
16 de octubre de 2025	Nueva característica Amazon DocumentDB 5.0 (parche de motor versión 3.0.15902) Amazon DocumentDB ahora admite el modo de doble pila (IPv4/IPv6) cuando el clúster se encuentra en una nube privada virtual (VPC). Para obtener más información, consulte esta publicación del blog y Amazon VPC y Amazon DocumentDB en la Guía para desarrolladores de Amazon DocumentDB.

Date	Notas de la versión
15 de septiembre de 2025	Correcciones de errores y otros cambios Amazon DocumentDB 5.0 (versión 3.0.17983 del parche del motor) Se corrigió un fallo del motor cuando las consultas incluyen una etapa posterior de `$replaceRoot` y `$lookup`.

Date	Notas de la versión
29 de julio de 2025	Nuevas características Amazon DocumentDB 5.0 (parche de motor, versión 3.0.15902) • Se agregó el soporte del operador de agregación de `$regexFindAll` . • Se lanzaron métricas de sobrecarga de índices y sobrecarga basada en colecciones. • Se agregaron CloudWatch métricas `AvailableMVCCIds` `LongestActiveGCRuntime` que proporcionan información sobre el estado general del clúster en lo que respecta a la recolección de basura y las recopilaciones anticuadas. • El `serverStatus` comando ahora incluye un campo «nvme_cache» que muestra el recuento de páginas escritas y no escritas en la caché de NVMe en las instancias. NVMe-backed • Se eliminó la limitación del número de campos para `$group` y `$project`. • Se añadió soporte para los siguientes operadores de cadenas: `$trim`, `$rtrim`, `$ltrim`, `$regexFindAll` , `$replaceOne` y `$replaceAll` . • Se añadió soporte para los siguientes operadores de matrices: `$first` y `$last`. • Se añadió soporte para el siguiente operador de etapas: `$collStats` . Correcciones de errores y otros cambios Amazon DocumentDB 5.0 (parche de motor versión 3.0.15902) Se corrigió un problema durante la creación que afectaba las colecciones grandes.

Date	Notas de la versión
28 de julio de 2025	Nuevas características Amazon DocumentDB 5.0 (parche de motor, versión 3.0.14228) Los clústeres basados en instancias de Amazon DocumentDB son compatibles con Amazon DocumentDB sin servidor, una configuración de escalado automático bajo demanda. Para obtener más información, consulte Uso de Amazon DocumentDB sin servidor y esta entrada de novedades del blog.
24 de marzo de 2025	Nueva característica Amazon DocumentDB 5.0 (versión 3.0.11800 del parche del motor) Se añadió soporte para los `postBatchResumeToken` en los flujos de cambios. Para obtener más información, consulte Reanudar un flujo de cambios con una publicación `BatchResumeToken` .

Date	Notas de la versión
15 de enero de 2025	Nuevas características Amazon DocumentDB 5.0 (versión 3.0.11800 del parche del motor) Amazon DocumentDB ahora mantiene la disponibilidad de lectura mediante el reinicio de las instancias de escritura. Las instancias de lectura ahora seguirán atendiendo las solicitudes de lectura cuando se reinicie la instancia de escritura. Correcciones de errores y otros cambios Amazon DocumentDB 5.0 (versión 3.0.11800 del parche del motor) • Se corrigió el comando `killOp` para gestionar un caso especial de inserciones masivas. • Uso mejorado de la I/O-related memoria de red en las instancias de Amazon DocumentDB. • Se corrigió el comando `count` para consultas de filtro `$text`.
18 de diciembre de 2024	Nuevas características Amazon DocumentDB 5.0 (versión de parche del motor 3.0.5721) Los clústeres basados en instancias de Amazon DocumentDB ahora admiten instancias R6GD. NVMe-backed Para obtener más información, consulte NVMe-backed instancias y esta entrada de novedades del blog.

Date	Notas de la versión
6 de noviembre de 2024	Nuevas características Amazon DocumentDB 5.0 (versión 3.0.11051 del parche del motor) • Se agregó soporte para las versiones mínimas de TLS `tls1.2+` y `tls1.3+` en el parámetro de clúster `tls`. • Se habilitó la compatibilidad con los caracteres dólar ($) y punto (.) en los nombres de campo. Para conocer las diferencias funcionales, consulte Dólar($) y punto(.) en los nombres de campos.
17 de septiembre de 2024	Nuevas características Amazon DocumentDB 5.0 (versión de parche del motor 3.0.10696) Amazon DocumentDB ahora admite la reconstrucción de índices con `reIndex` en `runCommand` . Para obtener más información, consulte Mantenimiento de índices mediante ReIndex . **Note** `reIndex` solo es compatible con Amazon DocumentDB 5.0 (versión de parche del motor 3.0.10696 y posterior). Correcciones de errores y otros cambios Amazon DocumentDB 5.0 (versión de parche del motor 3.0.10696) • `$setOnInsert` ahora admite el campo `_id` durante las insercion es resultantes de las operaciones de actualización. • Se corrigió un problema que impedía recuperar espacio de almacenamiento tras una actualización de una versión principal a Amazon DocumentDB 5.0.

Date	Notas de la versión
22 de agosto de 2024	Nueva característica Amazon DocumentDB 5.0 (todas las versiones de parches del motor) Los clústeres globales de Amazon DocumentDB ahora admiten transiciones de clústeres y conmutaciones por error de clústeres administradas. Para obtener más información, consulte Ejecución de una transición para un clúster global de Amazon DocumentDB y Ejecución de una conmutación por error administrada para un clúster global de Amazon DocumentDB. **Note** Las transiciones y conmutaciones por errores de clústeres globales solo son compatibles con Amazon DocumentDB 4.0 y 5.0.

<table>
<tr><th>Date</th><th>Notas de la versión</th></tr>
<tr><td>23 de julio de 2024</td><td>
Nuevas características

Amazon DocumentDB 5.0 (versión de parche del motor 3.0.8126)

• Se ha agregado compatibilidad con nombres de índice más largos (hasta 255 caracteres). Para obtener más información, consulte Restricciones en la nomenclatura
• Se ha duplicado el límite máximo de conexiones:
<table>
<tr><th>Tipo de instancia</th><th>Límite original</th><th>Nuevo límite</th></tr>
<tr><td>t3.medium</td><td>500</td><td>1 000</td></tr>
<tr><td>t4g.medium</td><td>500</td><td>1 000</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>

Date	Notas de la versión

Tipo de instancia	Límite original	Nuevo límite
r6g.8xlarge	28400	60000
r6g.12xlarge	30000	60000
r6g.16xlarge	30000	60000

Para obtener más información, consulte Cuotas de instancias.

Correcciones de errores y otros cambios

Amazon DocumentDB 5.0 (versión de parche del motor 3.0.8126)

Se ha mejorado la lógica para la sincronización de `CurrentTime` y `ResumeToken` para el flujo de cambios en los lectores.

Date	Notas de la versión

22 de julio de 2024	Nuevas características
	
	Amazon DocumentDB 5.0 (versión de parche del motor 3.0.6742)
	
	• Se ha agregado compatibilidad con el filtrado de auditoría DML. Ahora puede configurar las condiciones de filtrado para filtrar los registros de auditoría de DML en función de sus requisitos específicos en lugar de registrar todas las consultas de DML. Para obtener más información, consulte Filtrado de eventos de auditoría de DML.
	• Se ha agregado compatibilidad en la compresión de documentos para lo siguiente:
	  • Establecimiento de un umbral de compresión mínimo
	  • Habilitación de la compresión de las colecciones existentes (aplicable a los documentos nuevos)
	  • Permiso para la opción de compresión predeterminada en el nivel de clúster
	
	  Para obtener más información, consulte Gestión de la compresión de documentos a nivel de colección.
	• Se ha agregado compatibilidad para consumir flujos de cambios en las instancias de lectura. Para obtener más información, consulte Uso de flujos de cambios en instancias secundarias.

9 de julio de 2024	Nueva característica Para Amazon DocumentDB 5.0 (versión mínima del parche del motor 3.0.4780), las actualizaciones del certificado TLS CA ya no requieren reiniciar el sistema. Para obtener más información, consulte Actualización de los certificados TLS de Amazon DocumentDB.

Date	Notas de la versión
29 de mayo de 2024	Nuevas características Amazon DocumentDB 5.0 (versión de parche del motor 3.0.6742) • Se ha agregado compatibilidad para los operadores `$regexMatch` y `$regexFind`. • Se ha agregado compatibilidad para garantizar una precisión total en los registros de auditoría cuando se abordan números enteros grandes. Los registros de auditoría ahora mantienen la representación numérica exacta de todos los números, lo que evita cualquier pérdida de precisión.
3 de abril de 2024	Nuevas características Amazon DocumentDB 5.0 (versión de parche del motor 3.0.5721) • Se ha agregado compatibilidad con `bypassDocumentValidation` y un mensaje de error detallado para `$jsonSchema`. Para obtener más información acerca de `bypassDocumentValidation`, consulte circunvalación DocumentValidation. • Se ha agregado compatibilidad con `$expr`. • Se ha agregado compatibilidad con uniones no correlacionadas en `$lookup`. • Se ha agregado compatibilidad para retener las reglas de validación en la etapa de agregación de `$out`.

Date	Notas de la versión
10 de enero de 2024	Nuevas características Amazon DocumentDB 5.0 (versiones de parche del motor 3.0.4574, 3.0.4780 y 3.0.4960) • Se ha agregado compatibilidad para los índices vectoriales HNSW. Para obtener más información, consulte Búsqueda vectorial para Amazon DocumentDB. • Se ha agregado compatibilidad para índices parciales. Para obtener más información, consulte Índice parcial. • Se ha agregado compatibilidad para el tiempo de ejecución de GC en una colección dentro del comando `currentOp` . • Se ha agregado compatibilidad con índices de texto para la búsqueda de texto nativo en Amazon DocumentDB. Para obtener más información, consulte Búsquedas de texto con Amazon DocumentDB. • Se ha agregado compatibilidad con las palabras clave de esquema `$jsonSchema type`, `allOf`, `oneOf`, `anyOf`, `not`, `maxItems`, `minItems`, `maxProperties` , `minProperties` , `pattern`, `patternProperties` , `multipleOf` , `dependencies` y `uniqueItems` . Para obtener más información, consulte Cómo utilizar la validación de esquemas JSON. • Se ha agregado compatibilidad para los operadores aritméticos `$ceil`, `$floor`, `$ln`, `$log`, `$log10`, `$sqrt` y `$exp`. Para obtener más información, consulte Operadores aritméticos. • Se ha agregado compatibilidad para el operador de expresión condicional `$switch`. • Se ha agregado compatibilidad para compilaciones de índices de vector `IVFFLAT` en paralelo. La documentación se ha actualiza do al eliminar la limitación de compilaciones de índices vectoriales `IVFFLAT` paralelos de la guía para desarrolladores.

Date	Notas de la versión
	Correcciones de errores y otros cambios • Se ha agregado la funcionalidad que no distingue entre mayúsculas y minúsculas para invocar `db.runCommand("dbstats")` . Los clientes de Amazon DocumentDB 5.0 que tengan versiones de parches de motor anteriores a las 3.0.4960 o 2.0.10221 deben aplicar estos parches de motor más recientes. • Se ha corregido un error al invocar `db.coll.stats()` en la versión 1.7 y posteriores del intérprete de comandos de mongo. La documentación se ha actualizado al eliminar el consejo de solución de problemas `db.coll.stats()` del intérprete de comandos de mongo de la guía para desarrolladores.
29 de noviembre de 2023	Nuevas características Amazon DocumentDB 5.0 (versión de parche del motor 3.0.3727) Se ha agregado compatibilidad para la búsqueda vectorial. Para obtener más información, consulte esta publicación de blog y visite Búsqueda vectorial para Amazon DocumentDB en la Guía para desarrolladores de Amazon DocumentDB.
21 de noviembre de 2023	Nuevas características Amazon DocumentDB 5.0 (versión de parche del motor 3.0.3727) Se I/O-optimized ha añadido soporte para el almacenamiento. Para obtener más información, consulte Configuraciones de almacenamiento en clústeres de Amazon DocumentDB en la Guía para desarrolladores de Amazon DocumentDB. Se agregó la integración para el aprendizaje automático sin código con SageMaker Canvas. Para obtener más información, consulte No-code aprendizaje automático con Amazon SageMaker AI Canvas en la Guía para desarrolladores de Amazon DocumentDB.

Date	Notas de la versión
6 de noviembre de 2023	Amazon DocumentDB 5.0 (versión 3.0.3727 del parche del motor) Nuevas características • Se agregó compatibilidad con las palabras clave de esquema `$jsonSchema maxLength` , `minLength` , `maximum`, `minimum`, `exclusiveMaximum` , `exclusiveMinimum` , `items` y `additionalItems` . Tenga en cuenta que la validación del esquema JSON solo se admite en clústeres basados en instancias. • Se agregó soporte para el operador de canalización de agregació n `$convert` y sus operadores derivados de forma abreviada `$toBool`, `$toInt`, `$toLong`, `$toDouble` , `$toString` , `$toDecimal` , `$toObjectId` y `$toDate`. • Se agregó compatibilidad con los operadores de expresiones de conjuntos `$setDifference` , `$anyElementTrue` , y `$allElementTrue` . Correcciones de errores y otros cambios Se ha corregido un error que provocaba que no se mostrara de `-NaN` a `NaN` una actualización del flujo de cambios.
15 de septiembre de 2023	Nuevas características Amazon DocumentDB 5.0 (versión 3.0.3140 del parche del motor) Se ha agregado compatibilidad para el validador de esquemas $jsonSchema solo en clústeres basados en instancias. Para obtener más información, consulte Cómo utilizar la validación de esquemas JSON.

Date	Notas de la versión
13 de julio de 2023	Nuevas características Amazon DocumentDB 5.0 (versión de parche del motor 3.0.1948) • Se ha agregado compatibilidad con la compresión de documentos. • Se agregó compatibilidad para compilaciones de índices en paralelo. • Se agregó compatibilidad para el estado de compilación del índice. Correcciones de errores y otros cambios Amazon DocumentDB 5.0 (versión de parche del motor 3.0.1948) • Se ha corregido un problema de autenticación con `createCollection` los clústeres elásticos de Amazon DocumentDB cuando los usuarios no tienen acceso a las colecciones del sistema. • Se ha corregido un problema por el que las instancias regionales secundarias no podían usar los mismos nombres de instancia regionales principales.
7 de junio de 2023	Se han corregido errores y otros cambios Amazon DocumentDB 5.0 • Las instancias r5 y t3.medium ya son compatibles con Amazon DocumentDB 5.0. • `engineVersion` la opción predeterminada está `5.0.0` en el AWS SDK AWS CLI, y CloudFormation.

Date	Notas de la versión
10 de mayo de 2023	Correcciones de errores y otros cambios Amazon DocumentDB 5.0 (versión de parche del motor 3.0.1361) • Incorpora compatibilidad con los `ignoreunknownindex options` en la consola de createIndex. • Se dejó de añadir consultas de monitoreo interna a los registros de auditoría. • Los nombres de variables locales definidos por el usuario ahora admiten “_” (guion bajo) para operadores de proyección como `$let` y `$filter`.

Date	Notas de la versión

1 de marzo de 2023	Nuevas características
	
	Amazon DocumentDB 5.0 (versión de parche del motor 3.0.775)
	
	• Se ha presentado Amazon DocumentDB 5.0
	  • Compatibilidad con MongoDB 5.0 (soporte para los controladores API de MongoDB 5.0)
	  • Soporte para el cifrado a nivel de Client-side campo (FLE). Ahora puede cifrar los campos del lado del cliente antes de escribir los datos en el clúster de Amazon DocumentDB. Para obtener más información, consulte Cifrado a nivel de Client-side campo
	  • Nuevos operadores de agregación: `$dateAdd`, `$dateSubtract`
	• Se aumentó el límite de almacenamiento a 128 TiB para los clústeres de Amazon DocumentDB basados en instancias y para los clústeres elásticos basados en particiones.
	• Amazon DocumentDB 5.0 ahora admite escaneos de índices con el operador `$elemMatch` en el primer nivel de anidación. Los escaneos de índices se admiten cuando el filtro para solo consultas tiene un nivel de filtro de `$elemMatch` y la consulta `$elemMatch` anidada no admite el escaneo de índices.
	
	Forma de consulta que admite el escaneo de índices:
	
	`db.foo.find( { "a": {$elemMatch: { "b": "xyz", "c": "abc"} } })`
	
	Forma de consulta que no admite el escaneo de índices:
	
	`db.foo.find( { "a": {$elemMatch: { "b": {$elemMatch: { "d": "xyz", "e": "abc"} }} } })`

Amazon DocumentDB 4.0

Notas de la versión 4.0 del motor Amazon DocumentDB (compatibilidad con MongoDB 4.0).

Date	Notas de la versión
31 de agosto de 2026	Nueva característica Amazon DocumentDB ahora admite actualizaciones de versiones principales (MVU) locales directamente desde la versión 4.0 del motor a la versión 8.0. Esto elimina la necesidad de actualizar las versiones intermedias y le permite actualizar los clústeres de la versión 4.0 sin dejar de conservar los datos, las configuraciones y los ajustes del clúster existentes. Esta actualización está disponible en todas AWS las regiones en las que actualmente se admiten estas versiones. Para obtener más información, consulte Actualización local de la versión principal Amazon DocumentDB.
18 de agosto de 2026	Mejoras de seguridad y estabilidad Amazon DocumentDB 4.0 (versión del motor: 2.0.14287) [en curso] Este parche del motor proporciona las siguientes mejoras: • Procesamiento de solicitudes mejorado: Amazon DocumentDB ahora valida las solicitudes entrantes de los clientes de forma más exhaustiva a lo largo de la ruta de conexión y de análisis de mensajes. Esto mejora la resiliencia y la estabilidad en una gama más amplia de patrones de entrada. • Amazon DocumentDB, que refuerza la autenticación, ahora valida el intercambio de autenticación de forma más exhaustiva. Esto mejora la solidez a la hora de gestionar entradas inesperadas o con un formato incorrecto. • Access-control cumplimiento Amazon DocumentDB ahora verifica de manera uniforme los privilegios requeridos en todos los niveles de anidamiento para las operaciones de consulta anidadas.

Date	Notas de la versión
	Si no lo aplica antes, Amazon DocumentDB aplicará automáticamente este parche durante su próximo período de mantenimiento, 30 días después del lanzamiento.
27 de abril de 2026	Correcciones de errores y otros cambios Amazon DocumentDB 4.0 (versión del motor: 2.0.14278) Se trata de un parche de motor obligatorio que soluciona los siguiente s problemas: 1. Se ha corregido un problema que, en raras ocasiones, podía provocar que la recuperación puntual (PITR) y las actualizaciones de versiones principales (MVU) dejaran de responder. Todos los clústeres de Amazon DocumentDB 4.0 se actualizarán automáticamente a esta versión de parche durante el próximo período de mantenimiento.
9 de marzo de 2026	Correcciones de errores y otros cambios Amazon DocumentDB 4.0 (versión del motor: 2.0.14165) Esta actualización proporciona un parche de motor opcional con las siguientes mejoras: 1. La `DatabaseConnectionsActiveMax` métrica ahora hace un seguimiento `Connections (active)` de las instancias. Para obtener más información, consulta Cuotas de instancias. 2. Rendimiento mejorado de la autenticación SASL.

Date	Notas de la versión
22 de octubre de 2025	Nuevas características Amazon DocumentDB 4.0 (parche de motor, versión 2.0.11747) Se agregaron nuevos campos `collstats` para «mvccidStats» y «gcRuntimeStats» que proporcionan información sobre el estado general de la recolección en relación con la recolección de basura y la recolección anticuada.
16 de octubre de 2025	Nueva característica Amazon DocumentDB 4.0 (parche de motor, versión 2.0.11747) Amazon DocumentDB ahora admite el modo de doble pila (IPv4/IPv6) cuando el clúster se encuentra en una nube privada virtual (VPC). Para obtener más información, consulte esta publicación del blog y Amazon VPC y Amazon DocumentDB en la Guía para desarrolladores de Amazon DocumentDB.

Date	Notas de la versión
29 de julio de 2025	Nuevas características Amazon DocumentDB 4.0 (parche de motor, versión 2.0.11747) • Se lanzaron métricas de sobrecarga de índices y sobrecarga basada en colecciones. • Se agregaron CloudWatch métricas `AvailableMVCCIds` `LongestActiveGCRuntime` que proporcionan información sobre el estado general del clúster en lo que respecta a la recolecci ón de basura y las recopilaciones anticuadas. • Se añadió soporte para los siguientes operadores de cadenas: `$trim`, `$rtrim` y `$ltrim`. • Se añadió soporte para el siguiente operador de etapas: `$collStats` . Correcciones de errores y otros cambios Amazon DocumentDB 4.0 (parche de motor, versión 2.0.11747) Se corrigió un problema durante la creación que afectaba las colecciones grandes.
15 de enero de 2025	Correcciones de errores y otros cambios Amazon DocumentDB 4.0 (versión 2.0.11096 del parche de motor) • Se corrigió el comando `killOp` para gestionar un caso especial de inserciones masivas. • Uso mejorado de la I/O-related memoria de red en las instancias de Amazon DocumentDB.

Date	Notas de la versión
6 de noviembre de 2024	Nuevas características Amazon DocumentDB 4.0 (versión 2.0.10980 del parche del motor) • Se agregó soporte para las versiones mínimas de TLS `tls1.2+` y `tls1.3+` en el parámetro de clúster `tls`.
17 de septiembre de 2024	Correcciones de errores y otros cambios Amazon DocumentDB 4.0 (parche de motor, versión 2.0.10898) • `$setOnInsert` ahora admite el campo `_id` durante las insercion es resultantes de las operaciones de actualización. • Se corrigió un problema que impedía recuperar espacio de almacenamiento tras una actualización de una versión principal a Amazon DocumentDB 5.0.
22 de agosto de 2024	Nueva característica Amazon DocumentDB 4.0 (parche de motor, versión 2.0.5704) Los clústeres globales de Amazon DocumentDB ahora admiten transiciones de clústeres y conmutaciones por error de clústeres administradas. Para obtener más información, consulte Ejecución de una transición para un clúster global de Amazon DocumentDB y Ejecución de una conmutación por error administrada para un clúster global de Amazon DocumentDB. **Note** Las transiciones y conmutaciones por errores de clústeres globales solo son compatibles con Amazon DocumentDB 4.0 y 5.0.

<table>
<tr><th>Date</th><th>Notas de la versión</th></tr>
<tr><td>23 de julio de 2024</td><td>
Nuevas características

Amazon DocumentDB 4.0 (parche de motor, versión 2.0.10709)

• Se ha agregado compatibilidad con nombres de índice más largos (hasta 255 caracteres). Para obtener más información, consulte Restricciones en la nomenclatura
• Se ha duplicado el límite máximo de conexiones:
<table>
<tr><th>Tipo de instancia</th><th>Límite original</th><th>Nuevo límite</th></tr>
<tr><td>t3.medium</td><td>500</td><td>1 000</td></tr>
<tr><td>t4g.medium</td><td>500</td><td>1 000</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>Date</th><th>Notas de la versión</th></tr>
<tr><td></td><td>
<table>
<tr><th>Tipo de instancia</th><th>Límite original</th><th>Nuevo límite</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>
Para obtener más información, consulte Cuotas de instancias.
</td></tr>
<tr><td>22 de julio de 2024</td><td>Nuevas características

Amazon DocumentDB 4.0 (versión de parche del motor 2.0.10593)

• Se ha agregado compatibilidad con el filtrado de auditoría DML. Ahora puede configurar las condiciones de filtrado para filtrar los registros de auditoría de DML en función de sus requisitos específicos en lugar de registrar todas las consultas de DML. Para obtener más información, consulte Filtrado de eventos de auditoría de DML.</td></tr>
<tr><td>9 de julio de 2024</td><td>Nueva característica

Para Amazon DocumentDB 4.0 (versión mínima del parche del motor 2.0.10179), las actualizaciones del certificado TLS CA ya no requieren reiniciar el sistema. Para obtener más información, consulte Actualización de los certificados TLS de Amazon DocumentDB.</td></tr>
<tr><td>29 de mayo de 2024</td><td>Nuevas características

Amazon DocumentDB 4.0 (versión de parche del motor 2.0.10593)

• Se ha agregado compatibilidad para garantizar una precisión total en los registros de auditoría cuando se abordan números enteros grandes. Los registros de auditoría ahora mantienen la representación numérica exacta de todos los números, lo que evita cualquier pérdida de precisión.</td></tr>
</table>

Date	Notas de la versión
3 de abril de 2024	Nuevas características Amazon DocumentDB 4.0 (versión de parche del motor 2.0.10392) • Se ha agregado compatibilidad con `bypassDocumentValidation` para `$jsonSchema`. Para obtener más información acerca de `bypassDocumentValidation`, consulte circunvalación DocumentValidation. • Se ha agregado compatibilidad con `$expr`. • Se ha agregado compatibilidad con uniones no correlacionadas en `$lookup`. • Se ha agregado compatibilidad para retener las reglas de validación en la etapa de agregación de `$out`.
10 de enero de 2024	Nuevas características Amazon DocumentDB 4.0 (versiones de parche del motor 2.0.10124, 2.0.10179 y 2.0.10221) • Se ha agregado compatibilidad para el tiempo de ejecución de GC en una colección dentro del comando `currentOp`. • Se ha agregado compatibilidad con las palabras clave de esquema `$jsonSchema` `type`, `allOf`, `oneOf`, `anyOf`, `not`, `maxItems`, `minItems`, `maxProperties`, `minProperties`, `pattern`, `patternProperties`, `multipleOf`, `dependencies` y `uniqueItems`. Para obtener más información, consulte Cómo utilizar la validación de esquemas JSON. • Se ha agregado compatibilidad para los operadores aritméticos `$ceil`, `$floor`, `$ln`, `$log`, `$log10`, `$sqrt` y `$exp`. Para obtener más información, consulte Operadores aritméticos. • Se ha agregado compatibilidad para el operador de expresión condicional `$switch`.

Date	Notas de la versión
20 de diciembre de 2023	Otros cambios Se habilitó la compatibilidad con la actualización local de la versión principal en Amazon DocumentDB 4.0. Para obtener más informaci ón, consulte Actualización local de la versión principal Amazon DocumentDB.
6 de noviembre de 2023	Amazon DocumentDB 4.0 (versión 2.0.9876 del parche del motor) Nuevas características • Se agregó compatibilidad con las palabras clave de esquema `$jsonSchema maxLength` , `minLength` , `maximum`, `minimum`, `exclusiveMaximum` , `exclusiveMinimum` , `items` y `additionalItems` . Tenga en cuenta que la validación del esquema JSON solo se admite en clústeres basados en instancias. • Se agregó soporte para el operador de canalización de agregació n `$convert` y sus operadores derivados de forma abreviada `$toBool`, `$toInt`, `$toLong`, `$toDouble` , `$toString` , `$toDecimal` , `$toObjectId` y `$toDate`. • Se agregó compatibilidad con los operadores de expresiones de conjuntos `$setDifference` , `$anyElementTrue` , y `$allElementTrue` . Correcciones de errores y otros cambios Se ha corregido un error que provocaba que no se mostrara de `-NaN` a `NaN` una actualización del flujo de cambios.

Date	Notas de la versión
20 de septiembre de 2023	Nuevas características Se agregó soporte para las principales actualizaciones locales de las versiones principales en Amazon DocumentDB 4.0. Para obtener más información, consulte Actualización local de la versión principal Amazon DocumentDB.
15 de septiembre de 2023	Nuevas características Amazon DocumentDB 4.0 (versión 2.0.9686 del parche de motor) Se ha agregado compatibilidad para el validador de esquemas $jsonSchema solo en clústeres basados en instancias. Para obtener más información, consulte Cómo utilizar la validación de esquemas JSON.
13 de julio de 2023	Nuevas características Amazon DocumentDB 4.0 (versión de parche del motor 2.0.9259) • Se agregó compatibilidad para compilaciones de índices en paralelo. Correcciones de errores y otros cambios Amazon DocumentDB 4.0 (versión de parche del motor 2.0.9259) • Se dejó de añadir consultas de monitoreo interna a los registros de auditoría.

Date	Notas de la versión
4 de abril de 2023	Correcciones de errores y otros cambios Amazon DocumentDB 4.0 (versión de parche del motor 2.0.8934) • Se solucionó el problema con la auditoría de DML cuando estaba habilitada durante una carga de trabajo continua. • Se solucionó el problema con la auditoría de DML que provocaba que a los comandos agregados con una sugerencia se les pasara un valor de cadena. • Se solucionó el problema por el que el comando `listCollections` no funcionaba cuando los usuarios con el rol readwrite anydatabase tenían las opciones AuthorizedCollections y NameOnly configuradas en true. • Se solucionó el problema que impedía analizar correctamente la cadena numérica del nombre de un campo. • Cancele los cursores de larga duración cuando afecten a la recopilación de elementos no utilizados. • Los nombres de variables locales definidos por el usuario ahora admiten “_” (guion bajo) para operadores de proyección como `$let` y `$filter`.
27 de febrero de 2023	Correcciones de errores y otros cambios Amazon DocumentDB 4.0 Se ha añadido soporte para AWS Lambda. Para obtener más información, consulte Utilización AWS Lambda con Change Streams.

Date	Notas de la versión
9 de agosto de 2022	Nuevas características 4.0 • Se agregó soporte para el tipo de datos Decimal128. El Decimal128 es un tipo de datos BSON compatible con todas las regiones en las que DocumentDB está disponible. Para obtener más información, consulte Data Types. • Se ha añadido soporte para la auditoría de consultas de DML con Amazon Logs. CloudWatch Ahora Amazon DocumentDB puede registrar los eventos del lenguaje de manipulación de datos (DML) y del lenguaje de definición de datos (DDL) en Amazon Logs. CloudWatch Para obtener más información, consulte esta entrada del blog. Correcciones de errores y otros cambios 4.0 • Ahora puede cambiar su propia contraseña por una contraseña propia con privilegios `changeOwnPassword` .
25 de julio de 2022	Nuevas características Amazon DocumentDB 4.0 Ahora puede crear clústeres más rápido con la posibilidad de crear clones que usen el mismo volumen de clúster de DocumentDB y tengan los mismos datos que el clúster original. Para obtener más información, consulte Administración de clústeres de Amazon DocumentDB.

Date	Notas de la versión
27 de junio de 2022	Nuevas características Amazon DocumentDB 4.0 (versión de parche del motor 2.0.7509) Amazon DocumentDB cambia el tamaño de la base de datos de forma dinámica en función de los patrones de uso. Añadir más datos aumenta el espacio hasta 64 Tebibytes (TiB) y eliminar datos reduce el espacio asignado.
7 de abril de 2022	Nuevas características 4.0 (versiones 2.0.6142 y 2.0.6948 del parche del motor) Amazon DocumentDB Performance Insights ya está en vista previa. Ahora puede almacenar siete días del historial de rendimiento en un período continuo sin costo adicional. Para obtener más información, consulte Supervisión con información sobre rendimiento.

24 de enero de 2022	Nuevas características
	
	4.0 (versión 2.0.5170 del parche del motor)
	
	• Amazon DocumentDB ahora ofrece una prueba gratuita. Para obtener más información, consulte la página de prueba gratuita de Amazon DocumentDB.
	• Ahora puede utilizar funciones mejoradas con las consultas geoespaciales, incluidas las siguientes API:
	  • `$geoWithin`
	  • `$geoIntersects`
	• Se ha agregado compatibilidad con los siguientes operadores de MongoDB:
	  • `$mergeObjects`
	  • `$reduce`
	
	Para obtener más información, consulte Consulta de datos geoespaciales con Amazon DocumentDB.

Date	Notas de la versión

21 de enero de 2022	Nuevas características
	
	Amazon DocumentDB 4.0 (versión de parche del motor 2.0.5706)
	
	• Ahora se admiten las instancias Graviton2 (r6g.large, r6g.2xlarge, r6g.4xlarge, r6g.8xlarge, r6g.12xlarge, r6g.16xlarge y t4g.medium) de Amazon DocumentDB.
	
	Nuevas características
	
	4.0 (versión 2.0.5706 del parche del motor)
	
	• Añadida compatibilidad para las siguientes API de MongoDB:
	  • `$reduce`
	  • `$mergeObjects`
	  • `$geoWithin`
	  • `$geoIntersects`

Date	Notas de la versión

25 de octubre de 2021	Nuevas características
	
	4.0 (versión 2.0.5704 del parche del motor)
	
	• Añadida compatibilidad para las siguientes API de MongoDB
	  • `$literal`
	  • `$map`
	  • `$$ROOT`
	• Soporte para capacidades de consulta. GeoSpatial Consulte esta publicación en el blog para obtener más información
	• Compatibilidad para el control de acceso con funciones definidas por el usuario. Consulte esta publicación en el blog para obtener más información
	• Controlador JDBC de Amazon DocumentDB para permitir la conectividad desde herramientas de BI como Tableau y herramien tas de consulta como SQL Workbench
	
	Correcciones de errores y otros cambios
	
	4.0 (versión 2.0.5704 del parche del motor)
	
	• Se corrigió un error `$natural` para ordenar correctamente cuando había un `.sort()` explícito junto con `$natural`
	• Se corrigió un error para que el flujo de cambios funcionara con `$redact`
	• Se corrigió un error para que `$ifNull` funcionara con una matriz vacía
	• Se ha corregido un error relacionado con la pérdida consumption/ server excesiva de recursos cuando se elimina a un usuario que ya ha iniciado sesión o se le revoca el privilegio de ese usuario para realizar una actividad en curso
	• Se corrigió un error en `listDatabase` y comprobación de privilegios `listCollection`

Date	Notas de la versión
	• Se corrigió un error: lógica de deduplicación para elementos de varias claves
24 de junio de 2021	Nuevas características 4.0 (versión 2.0.3371 del parche de motor) • Ahora se admiten las instancias r5.8xlarge y r5.16xlarge. Obtenga más información en la publicación en el blog Amazon DocumentDB ahora es compatible con instancias r5.8xlarge y r5.16xlarge. • Ahora se admiten clústeres globales para proporcionar recuperación ante desastres en caso de Region-wide interrupciones y permitir lecturas globales de baja latencia al permitir la lectura desde el clúster de Amazon DocumentDB más cercano. Tenga en cuenta que los clústeres globales no se admiten en las siguientes regiones: América del Sur (São Paulo), Europa (Milán), China (Pekín) y China (Ningxia).

Date	Notas de la versión
4 de mayo de 2021	Nuevas características 4.0 (versión 2.0.3371 del parche del motor) • `renameCollection` • `$zip` • `$indexOfArray` • `$reverseArray` • `$natural` • Compatibilidad `$hint` para actualizaciones • Escaneo de índice para `distinct` Correcciones de errores y otros cambios 4.0 (versión 2.0.3371 del parche del motor) • Uso de memoria reducido para consultas `$in` • Se corrigió una pérdida de memoria en los índices de varias claves • Se corrigió el plan de explicación y el resultado del generador de perfiles para `$out` • Se agregó un tiempo de espera para las operaciones del sistema de monitoreo interno para mejorar la fiabilidad • Se corrigió un defecto que afectaba a los predicados de consulta pasados a índices de varias claves
15 de enero de 2021	Nuevas características Amazon DocumentDB 4.0 (versión de parche del motor 2.0.722) • Ninguno

Date	Notas de la versión
9 de noviembre de 2020	Nuevas características Amazon DocumentDB 4.0 (versión de parche del motor 2.0.722) • Compatibilidad con MongoDB 4.0 • Transacciones ACID • Compatibilidad para `cluster(client.watch()` o `mongo.watch())` y flujos de cambios `(db.watch())` a nivel de la base de datos • Posibilidad de iniciar o reanudar un flujo de cambios mediante `startAtOperationTime` • Ampliación del período de retención del flujo de cambios a 7 días (antes era de 24 horas) • AWS DMS objetivo para Amazon DocumentDB 4.0 • CloudWatch métricas:`TransactionsOpen` ,`TransactionsOpenMax` , `TransactionsAborted` `TransactionsStarted` , y `TransactionsCommitted` • Nuevos campos para transacciones en `currentOp` , `ServerStatus` , y `profiler`. • Posibilidad de utilizar un índice con la etapa de agregación `$lookup` • Las consultas `find()` con proyecciones se pueden atender directamente desde un índice (consulta cubierta) • Posibilidad de usar `hint()` con `findAndModify` • Optimizaciones de rendimiento para el operador `$addToSet` • Mejoras para reducir los tamaños de los índices generales. • Nuevos operadores de agregación: `$ifNull`, `$replaceRoot` , `$setIsSubset` , `$setInstersection` , `$setUnion` , y `$setEquals` • Con los comandos `ListCollection` y `ListDatabase` , ahora puede usar opcionalmente los parámetros `authorizedCollections` y `authorizedDatabases` para permitir a

Date	Notas de la versión
	los usuarios enumerar las colecciones y bases de datos a las que tienen permiso de acceso sin necesitar las funciones `listCollections` y `listDatabase`, respectivamente. • Los usuarios también pueden finalizar sus propios cursores sin necesidad del rol `KillCursor` • La comparación de tipos numéricos de subdocumentos ya es coherente con la comparación de tipos numéricos de documentos de primer nivel. Amazon DocumentDB ya es compatible con la versión 4.0 de MongoDB. Correcciones de errores y otros cambios Amazon DocumentDB 4.0 (versión de parche del motor 2.0.722) • `$setOnInsert` ya no permite actualizaciones cuando se utiliza el operador posicional $. Amazon DocumentDB ya es compatible con la versión 4.0 de MongoDB. • Se solucionó el problema con `$createCollection` y conjunto `autoIndexId` • Proyección para documentos anidados • Se modificó la configuración predeterminada de la memoria de trabajo para que se escale con el tamaño de la memoria de la instancia • Mejoras en la recopilación de elementos no utilizados • Búsqueda con una clave vacía en la ruta, diferencia de comportamiento con mongo • Se corrigió un error `dateToString` en el comportamiento de la zona horaria • Se corrigió `$push` (agregación) para respetar el orden de clasificación • Se corrigió un error en `$currentOp` con agregado • Se solucionó un problema de `readPreference` en secundario

Date	Notas de la versión
	• Se solucionó el problema al validar `$createIndex` en la misma base de datos en la que se ejecutó el comando • Se corrigió un comportamiento incoherente para `minKey`, la búsqueda `maxKey` falla • Se corrigió el problema por el que el operador `$size` no funcionaba con una matriz compuesta • Se solucionó el problema de la negación de `$in` con regex • Se corrigió un error que provocaba que el comando `$distinct` se ejecutara contra una vista • Se corrigió un problema que provocaba que los comandos de agregación y búsqueda clasificaran los campos que faltan de forma diferente • Se corrigió el `$eq` en la expresión regular que no verificaba el tipo • Se corrigió un error `$currentDate` en el comportamiento de la posición ordinal de la marca temporal • Se fijó la granularidad de milisegundos para `$currentDate`

Amazon DocumentDB 3.6

Notas de versión de la versión 3.6 del motor Amazon DocumentDB (compatibilidad con MongoDB 3.6).

Date	Notas de la versión
31 de agosto de 2026	Nueva característica Amazon DocumentDB ahora admite actualizaciones de versiones principales (MVU) locales directamente desde la versión 3.6 del motor a la versión 8.0. Esto elimina la necesidad de actualizar las versiones intermedias y le permite actualizar los clústeres de la versión 3.6, conservando al mismo tiempo los datos, las configuraciones y los ajustes del clúster existentes. Esta actualización está disponible en todas AWS las regiones en las que actualmente se admiten estas

Date	Notas de la versión
	versiones. Para obtener más información, consulte Actualización local de la versión principal Amazon DocumentDB.
18 de agosto de 2026	Mejoras de seguridad y estabilidad Amazon DocumentDB 3.6 (versión del motor: 1.0.208702) [en curso] Este parche del motor proporciona las siguientes mejoras: • Procesamiento de solicitudes mejorado: Amazon DocumentDB ahora valida las solicitudes entrantes de los clientes de forma más exhaustiva a lo largo de la ruta de conexión y de análisis de mensajes. Esto mejora la resiliencia y la estabilidad en una gama más amplia de patrones de entrada. • Amazon DocumentDB, que refuerza la autenticación, ahora valida el intercambio de autenticación de forma más exhaustiva. Esto mejora la solidez a la hora de gestionar entradas inesperadas o con un formato incorrecto. Si no lo aplica antes, Amazon DocumentDB aplicará automáticamente este parche durante el siguiente período de mantenimiento, 30 días después del lanzamiento.
27 de abril de 2026	Nuevas características Amazon DocumentDB 3.6 (versión del motor: 1.0.208692) Esta actualización proporciona un parche de motor opcional con las siguientes mejoras: 1. Se añadió soporte para los `startAtOperationTime` en los flujos de cambios.

Date	Notas de la versión
12 de febrero de 2026	Correcciones de errores y otros cambios Amazon DocumentDB 3.6 (versión del motor: 1.0.208675) 1. La `DatabaseConnectionsActiveMax` métrica ahora registra las `Connections (active)` instancias. Para obtener más información, consulta Cuotas de instancias. 2. Rendimiento mejorado de la autenticación SASL. 3. Se ha corregido un error en el comando $indexStats.
20 de agosto de 2024	Nueva característica Para Amazon DocumentDB 3.6 (versión mínima del parche del motor 1.0.208662), las actualizaciones del certificado TLS CA ya no requieren reiniciar el sistema. Para obtener más información, consulte Actualización de los certificados TLS de Amazon DocumentDB.
22 de julio de 2024	Nuevas características Amazon DocumentDB 3.6 (versión de parche del motor 1.0.208662) Se han eliminado las limitaciones de índice en las instancias db.r5.* y db.r6.* en la MVU de Amazon DocumentDB. Para obtener más información, consulte Requisitos previos. Correcciones de errores y otros cambios Amazon DocumentDB 3.6 (versión de parche del motor 1.0.208662) Amazon DocumentDB ahora reconoce -NaN como un token JSON válido.
20 de diciembre de 2023	Otros cambios Se habilitó la compatibilidad con la actualización local de la versión principal en Amazon DocumentDB 3.6. Para obtener más informaci ón, consulte Actualización local de la versión principal Amazon DocumentDB.

Date	Notas de la versión
17 de noviembre de 2023	Correcciones de errores y otros cambios Amazon DocumentDB 3.6 (versión de parche del motor 1.0.208570) Los nombres de variables locales definidos por el usuario ahora admiten "_" (guion bajo) para operadores de proyección como `$let` y `$filter`.
20 de septiembre de 2023	Nuevas características Se agregó soporte para las principales actualizaciones locales de las versiones principales en Amazon DocumentDB 3.6. Para obtener más información, consulte Actualización local de la versión principal Amazon DocumentDB.
2 de febrero de 2023	Correcciones de errores y otros cambios Amazon DocumentDB 3.6 (versión de parche del motor 1.0.208432) • Se solucionó el problema con la auditoría de DML cuando estaba habilitada durante una carga de trabajo continua. • Se solucionó el problema con la auditoría de DML que provocaba que a los comandos agregados con una sugerencia se les pasara un valor de cadena. • Se solucionó el problema por el que el comando `listCollections` no funcionaba cuando los usuarios con el rol readwrite anydatabase tenían las opciones AuthorizedCollections y NameOnly configuradas en true. • Se solucionó el problema que impedía analizar correctamente la cadena numérica del nombre de un campo. • Cancele los cursores de larga duración cuando afecten a la recopilación de elementos no utilizados.

Date	Notas de la versión
9 de agosto de 2022	Nuevas características Amazon DocumentDB 3.6 (parche de motor, versión 1.0.208152) • Se agregó soporte para el tipo de datos Decimal128. El Decimal128 es un tipo de datos BSON compatible con todas las regiones en las que DocumentDB está disponible. Para obtener más información, consulte Data Types. • Se ha añadido soporte para la auditoría de consultas de DML con Amazon Logs. CloudWatch Ahora Amazon DocumentDB puede registrar eventos del lenguaje de manipulación de datos (DML) y eventos del lenguaje de definición de datos (DDL) en Amazon Logs. CloudWatch Para obtener más información, consulte esta entrada del blog. Correcciones de errores y otros cambios Amazon DocumentDB 3.6 (parche de motor, versión 1.0.208152) • Ahora puede cambiar su propia contraseña por una contraseña propia con privilegios `changeOwnPassword` .
7 de abril de 2022	Nuevas características Amazon DocumentDB 3.6 (versiones del parche del motor 1.0.207836 y 1.0.208015) Amazon DocumentDB Performance Insights ya está en vista previa. Ahora puede almacenar siete días del historial de rendimiento en un período continuo sin costo adicional. Para obtener más información, consulte Supervisión con información sobre rendimiento.

Date	Notas de la versión

24 de enero de 2022	Nuevas características
	
	Amazon DocumentDB 3.6 (parche de motor, versión 1.0.207684)
	
	• Amazon DocumentDB ahora ofrece una prueba gratuita. Para obtener más información, consulte la página de prueba gratuita de Amazon DocumentDB.
	• Ahora puede utilizar funciones mejoradas con las consultas geoespaciales, incluidas las siguientes API:
	  • `$geoWithin`
	  • `$geoIntersects`
	• Se ha agregado compatibilidad con los siguientes operadores de MongoDB:
	  • `$mergeObjects`
	  • `$reduce`
	
	Para obtener más información, consulte Consulta de datos geoespaciales con Amazon DocumentDB.

21 de enero de 2022	Nuevas características
	
	Amazon DocumentDB 3.6 (parche de motor, versión 1.0.207781)
	
	• Añadida compatibilidad para las siguientes API de MongoDB:
	  • `$reduce`
	  • `$mergeObjects`
	  • `$geoWithin`
	  • `$geoIntersects`

Date	Notas de la versión

25 de octubre de 2021	Nuevas características
	
	Amazon DocumentDB 3.6 (parche de motor, versión 1.0.207780)
	
	• Añadida compatibilidad para las siguientes API de MongoDB
	  • `$literal`
	  • `$map`
	  • `$$ROOT`
	• Soporte para capacidades de consulta. GeoSpatial Consulte esta publicación en el blog para obtener más información
	• Compatibilidad para el control de acceso con funciones definidas por el usuario. Consulte esta publicación en el blog para obtener más información
	• Controlador JDBC de Amazon DocumentDB para permitir la conectividad desde herramientas de BI como Tableau y herramien tas de consulta como SQL Workbench
	
	Correcciones de errores y otros cambios
	
	Amazon DocumentDB 3.6 (versión 1.0.207780 del parche del motor)
	
	• Se corrigió un error `$natural` para ordenar correctamente cuando había un `.sort()` explícito junto con `$natural`
	• Se corrigió un error para que el flujo de cambios funcionara con `$redact`
	• Se corrigió un error para que `$ifNull` funcionara con una matriz vacía
	• Se ha corregido un error relacionado con la pérdida excesiva consumption/server de recursos cuando se elimina a un usuario que actualmente ha iniciado sesión o se revoca el privilegio de ese usuario para realizar una actividad en curso
	• Se corrigió un error en `listDatabase` y comprobación de privilegios `listCollection`

Date	Notas de la versión
	• Se corrigió un error: lógica de deduplicación para elementos de varias claves
24 de junio de 2021	Nuevas características Amazon DocumentDB 3.6 (versión 1.0.207117 del parche del motor) • Ahora se admiten las instancias r5.8xlarge y r5.16xlarge. Obtenga más información en la publicación en el blog Amazon DocumentDB ahora es compatible con instancias r5.8xlarge y r5.16xlarge. • Ahora se admiten clústeres globales para proporcionar recuperación ante desastres en caso de Region-wide interrupciones y permitir lecturas globales de baja latencia al permitir la lectura desde el clúster de Amazon DocumentDB más cercano. Tenga en cuenta que los clústeres globales no se admiten en las siguientes regiones: América del Sur (São Paulo), Europa (Milán), China (Pekín) y China (Ningxia).

Date	Notas de la versión
4 de mayo de 2021	Nuevas características Amazon DocumentDB 3.6 (versión 1.0.207117 del parche del motor) • `renameCollection` • `$zip` • `$indexOfArray` • `$reverseArray` • `$natural` • Compatibilidad `$hint` para actualizaciones • Escaneo de índice para `distinct` Correcciones de errores y otros cambios Amazon DocumentDB 3.6 (parche de motor, versión 1.0.207117) • Uso de memoria reducido para consultas `$in` • Se corrigió una pérdida de memoria en los índices de varias claves • Se corrigió el plan de explicación y el resultado del generador de perfiles para `$out` • Se agregó un tiempo de espera para las operaciones del sistema de monitoreo interno para mejorar la fiabilidad • Se corrigió un defecto que afectaba a los predicados de consulta pasados a índices de varias claves

Date	Notas de la versión
15 de enero de 2021	Nuevas características Amazon DocumentDB 3.6 (versión de parche del motor 1.0.206295) • Posibilidad de utilizar un índice con la etapa de agregación `$lookup` • Las consultas `find()` con proyecciones se pueden atender directamente desde un índice (consulta cubierta) • Posibilidad de usar `hint()` con `findAndModify` • Optimizaciones de rendimiento para el operador `$addToSet` • Mejoras para reducir el tamaño general de los índices • Nuevos operadores de agregación: `$ifNull`, `$replaceRoot` , `$setIsSubset` , `$setInstersection` , `$setUnion` , y `$setEquals` • Los usuarios también pueden finalizar sus propios cursores sin necesidad del rol `KillCursor`
9 de noviembre de 2020	Nuevas características Amazon DocumentDB 3.6 (versión de parche del motor 1.0.206295) • Ninguno Correcciones de errores y otros cambios Amazon DocumentDB 3.6 (versión de parche del motor 1.0.206295) • Ninguno

Date	Notas de la versión
30 de octubre de 2020	Nuevas características Amazon DocumentDB 3.6 (versión de parche del motor 1.0.206295) • Se agregó la posibilidad de abrir un cursor de flujo de cambios en el nivel del clúster (`client.watch()` o `mongo.watch()`) y la base de datos (`db.watch()`) • Capacidad de ampliar el período de retención del flujo de cambios a 7 días (antes era de 24 horas) Correcciones de errores y otros cambios Amazon DocumentDB 3.6 (versión de parche del motor 1.0.206295) • Varias mejoras generales en el rendimiento de los casos • Una mejora de seguridad específica • Se ha corregido un problema por el que se podía omitir la clasificación en el segundo campo de un índice compuesto • Habilite el índice normal para garantizar la igualdad en un solo campo de un índice de varias claves (no compuesto) • Condición de carrera de autenticación fija • Se ha corregido un problema que provocaba un bloqueo poco frecuente en la recopilación de elementos no utilizados • Mejoras en la seguridad RBAC • Añadida la métrica `databaseConnectionsMax` • Mejoras en el rendimiento de determinadas cargas de trabajo en las instancias `r5.24xlarge`

Date	Notas de la versión
22 de septiembre de 2020	Nuevas características Amazon DocumentDB 3.6 (versión de parche del motor 1.0.206295) • Etapa de agregación `$out` • Aumentó hasta 10 veces el número máximo de conexiones y de cursores por instancia Correcciones de errores y otros cambios Amazon DocumentDB 3.6 (versión de parche del motor 1.0.206295) • Ninguno
10 de julio de 2020	Nuevas características Amazon DocumentDB 3.6 (versión de parche del motor 1.0.206295) • Copia de instantáneas entre regiones Correcciones de errores y otros cambios Amazon DocumentDB 3.6 (versión de parche del motor 1.0.206295) • Ninguno

Date	Notas de la versión
30 de junio de 2020	Nuevas características Amazon DocumentDB 3.6 (versión de parche del motor 1.0.206295) • Instancias medianas T3 Correcciones de errores y otros cambios Amazon DocumentDB 3.6 (versión de parche del motor 1.0.206295) • Recuperación de memoria inactiva para instancias t3 • Mejoras en la autenticación • Rendimiento de autenticación de SASL mejorado • Se solucionó el problema `currentOp` que se producía al superar el máximo de operaciones • Se solucionó el problema `killOps` de actualización y eliminación masivas • Mejoras de rendimiento de `$sample` con `$match` • Se ha corregido la compatibilidad para `$$` en el segundo caso en la fase de redacción • Se corrigieron varias causas subyacentes de los bloqueos recurrentes • Mejoras en el barrido de TTL para reducir el iOS y la latencia • Utilización optimizada de la memoria para `$unwind` • Se corrigió la condición de carrera de estadísticas de colección fija • Se corrigió la condición de carrera durante la compilación simultánea del índice • Se corrigió un bloqueo infrecuente en `hash_search` en índice

Notas generales de la versión de Amazon DocumentDB

Actualizaciones generales de los servicios, que incluyen la disponibilidad regional, las integraciones de herramientas y las funciones entre versiones.

Date	Notas de la versión
15 de septiembre de 2026	Nueva característica El tipo de instancia sin servidor de Amazon DocumentDB ya está disponible en las versiones 5.0 y 8.0 de los motores Amazon DocumentDB en las siguientes regiones: • Asia-Pacífico (Osaka) • Asia-Pacífico (Yakarta) • Asia-Pacífico (Malasia) • Asia-Pacífico (Melbourne) • Asia-Pacífico (Tailandia) • Oeste de Canadá (Calgary) • Europa (Zúrich) • Europa (Estocolmo) • Israel (Tel Aviv) • México (centro) Para obtener más información, consulte Clases de instancias admitidas por región y Uso de Amazon DocumentDB sin servidor.
29 de abril de 2026	Nueva característica Amazon DocumentDB ya está disponible en la región Canadá Oeste (Calgary). Para obtener más información, consulte esta entrada del blog.
17 de abril de 2026	Nueva característica Ya está disponible un escalado más rápido en DocumentDB sin servidor.
11 de febrero de 2026	Nueva característica

Date	Notas de la versión
	Amazon DocumentDB ya está disponible en la región Asia-Pacífico (Melbourne). Para obtener más información, consulte esta entrada del blog. Amazon DocumentDB ya está disponible en la región de Europa (Zúrich). Para obtener más información, consulte esta entrada del blog.
08 de enero de 2026	Nueva característica Amazon DocumentDB ya está disponible en la región Asia-Pacífico (Yakarta). Para obtener más información, consulte esta entrada del blog.
13 de octubre de 2025	Nueva característica Amazon DocumentDB ahora admite instancias R6G y T4G en la región de Asia Pacífico (Hyderabad). Para obtener más información, consulte Clases de instancias admitidas por región.
7 de octubre de 2025	Nueva característica Amazon DocumentDB ya está disponible en las siguientes regiones: • Asia-Pacífico (Malasia) • Asia-Pacífico (Osaka) • Asia-Pacífico (Tailandia) • México (centro) Para obtener más información, consulte esta entrada del blog.

Date	Notas de la versión
26 de septiembre de 2025	Nuevas características Amazon DocumentDB ahora admite la copia y el uso compartido de instantáneas entre regiones y entre cuentas. Para obtener más información, consulte Copiar instantáneas de un lado a otro Regiones de AWS y Uso compartido de instantáneas de clústeres de Amazon DocumentDB.
18 de junio de 2025	Nueva característica Amazon DocumentDB ya está disponible en la región de Israel (Tel Aviv).
8 de mayo de 2025	Nueva característica Amazon DocumentDB ya está disponible en la región de Europa (Estocolmo). Para obtener más información, consulte esta entrada del blog.
2 de abril de 2025	Correcciones de errores y otros cambios Se corrigió un error en la creación de índices vectoriales con índices en segundo plano.
6 de febrero de 2025	Nueva característica Amazon DocumentDB ahora está integrado con AWS Toolkit for Visual Studio Code. Para obtener más información, consulte esta publicación de novedades del blog y consulte Uso de Amazon DocumentDB en el kit de herramientas en la Guía del usuario de AWS Toolkit for Visual Studio Code .

Date	Notas de la versión
22 de octubre de 2024	Nueva característica Amazon DocumentDB ahora admite métricas de rendimiento de la red de almacenamiento `StorageNetworkReceiveThroughput` , `StorageNetworkTransmitThroughput` y `StorageNetworkThroughput` . Para obtener más información, consulte Evaluación del uso de las instancias de Amazon DocumentDB con métricas CloudWatch.
18 de septiembre de 2024	Nueva característica Amazon DocumentDB ya está disponible en la región de África (Ciudad del Cabo). Para obtener más información, consulte esta entrada del blog. Amazon DocumentDB ya está disponible en la región de Europa (España). Para obtener más información, consulte esta entrada del blog.
25 de junio de 2024	Nueva característica La autenticación con ARN de AWS IAM está disponible en los clústeres 5.0 basados en instancias de Amazon DocumentDB en todas las regiones compatibles. Para obtener más información, consulte Autenticación mediante la identidad de IAM.
3 de abril de 2024	Nueva característica Amazon DocumentDB ya está disponible en la región de Oriente Medio (Emiratos Árabes Unidos). Para obtener más información, consulte esta entrada del blog. Correcciones de errores y otros cambios • Se ha corregido un error al invocar `db.coll.stats()` en la versión 1.7 y posteriores del intérprete de comandos de mongo. • Se ha corregido un problema de pérdida de memoria en las consultas de flujo de cambios que contenían `$regex` como parte de la misma canalización de agregación.

Date	Notas de la versión
30 de enero de 2024	Nuevas características Clústeres globales de Amazon DocumentDB Los clústeres globales ahora están disponibles en ambas AWS GovCloud (US) regiones: AWS GovCloud (US-East) y AWS GovCloud (US-West).
13 de diciembre de 2023	Nuevas características Se ha agregado compatibilidad con la conectividad de EC2 con un solo clic. Para obtener más información, consulte Conectarse mediante Amazon EC2.
17 de noviembre de 2023	Nuevas características Amazon DocumentDB ya está disponible en la región AWS GovCloud (US-East). Para obtener más información, consulte esta entrada del blog.
25 de septiembre de 2023	Nuevas características Amazon DocumentDB ya está disponible en la región Asia-Pacífico (Hong Kong). Para obtener más información, consulte esta entrada del blog.
11 de septiembre de 2023	Nuevas características Amazon DocumentDB ya está disponible en la región Asia-Pacífico (Hyderabad). Para obtener más información, consulte esta entrada del blog.
29 de abril de 2022	Nuevas características Amazon DocumentDB ya está disponible en la región de China (Pekín). Para obtener más información, consulte esta entrada del blog.

Date	Notas de la versión
16 de marzo de 2022	Nuevas características Amazon DocumentDB ya está disponible en la región de Europa (Milán). Para obtener más información, consulte esta entrada del blog.
8 de febrero de 2022	Nuevas características Las instancias R6g y T4g de Amazon DocumentDB ya están disponibles en Asia Pacífico, Sudamérica y Europa. Para obtener más información, consulte esta entrada del blog.
4 de mayo de 2021	Nuevas características Consulte todas las nuevas características en esta publicación en el blog.
9 de noviembre de 2020	Nuevas características Consulte todas las nuevas características en esta publicación en el blog.
30 de octubre de 2020	Nuevas características Consulte todas las nuevas características en esta publicación en el blog.
22 de septiembre de 2020	Nuevas características Consulte todas las nuevas características en esta publicación en el blog.
10 de julio de 2020	Nuevas características Consulte todas las nuevas características en esta publicación en el blog.
30 de junio de 2020	Nuevas características Consulte todas las nuevas características en esta publicación en el blog.

Notas de la versión de Amazon DocumentDB Elastic Clusters

Notas de versión sobre los clústeres elásticos de Amazon DocumentDB.

Date	Notas de la versión
28 de enero de 2025	Nueva característica Para ofrecer conectividad con un solo clic, Amazon DocumentDB ahora está integrado con clústeres elásticos y AWS CloudShell basados en instancias. Para obtener más información, consulte cualquiera de los siguientes temas o todos ellos: • Introducción a Amazon DocumentDB • Introducción a los clústeres elásticos de Amazon DocumentDB • Publicación de novedades del blog • Entrada de la guía básica técnica
12 de noviembre de 2024	Nuevas características Los clústeres elásticos de Amazon DocumentDB son compatibles con los índices en segundo plano.
1 de noviembre de 2024	Nueva característica Amazon DocumentDB ahora admite acciones de mantenimiento de clústeres elásticos. Para obtener más información, consulte Mantenimiento de clústeres elásticos de Amazon DocumentDB.
8 de agosto de 2024	Nueva característica Los clústeres elásticos de Amazon DocumentDB ya están disponibles en las regiones de Asia Pacífico (Hong Kong), Canadá (Central) y Europa (París). Para obtener más información, consulte Disponibilidad regional para clústeres elásticos.
8 de julio de 2024	Nueva característica Los clústeres elásticos de Amazon DocumentDB ya están disponibles en la región de Europa (Milán). Para obtener más información, consulte Disponibilidad regional para clústeres elásticos.

Date	Notas de la versión
22 de febrero de 2024	Nuevas características Clústeres elásticos de Amazon DocumentDB Los clústeres elásticos de Amazon DocumentDB ahora admiten las siguientes características: • Réplicas legibles de instancias con particiones secundarias: para obtener más información, consulte el paso 5b de Paso 1: crear un clúster elástico. • Start/stop clúster: para obtener más información, consulteDetención e inicio de un clúster elástico de Amazon DocumentDB. • Instancias con particiones configurables: para obtener más informaci ón, consulte el paso 5b de Paso 1: crear un clúster elástico. • Copias de seguridad automáticas para instantáneas: para obtener más información, consulte Administración de una copia de seguridad automática de una instantánea de un clúster elástico. • Copia de instantáneas: para obtener más información, consulte Copia de una instantánea de un clúster elástico.
30 de enero de 2024	Nuevas características Clústeres elásticos de Amazon DocumentDB Los clústeres elásticos de Amazon DocumentDB ya están disponibles en las siguientes regiones: • Asia-Pacífico (Mumbai) • Asia-Pacífico (Seúl) • América del Sur (São Paulo) • Europa (Londres) Para obtener más información, consulte Disponibilidad regional y de versión del clúster elástico.

Date	Notas de la versión

3 de agosto de 2023	Nuevas características
	
	Clústeres elásticos de Amazon DocumentDB
	
	• Los clústeres elásticos de Amazon DocumentDB ahora admiten las siguientes operaciones:
	  • `top`
	  • `collStats`
	  • `hint`
	  • `dataSize`
	
	  Consulte API, operaciones y tipos de datos de MongoDB admitidos en Amazon DocumentDB para ver una lista completa de comandos y operaciones admitidos.
	• Ahora se admiten los índices de tiempo de vida (TTL, Tiempo de vida).
	• Las `hints` de índices son compatibles con las expresiones de índice.

22 de marzo de 2023	Nuevas características Los clústeres elásticos de Amazon DocumentDB ya están disponibles en las regiones de Asia Pacífico (Singapur), Asia Pacífico (Sídney) y Asia Pacífico (Tokio). Para obtener más información, consulte Disponibilidad regional y de versión del clúster elástico.

Date	Notas de la versión
30 de noviembre de 2022	Nuevas características Clústeres elásticos de Amazon DocumentDB Los clústeres elásticos de Amazon DocumentDB son un nuevo tipo de clúster de Amazon DocumentDB que permite a los usuarios aprovechar las API de partición de MongoDB para escalar horizontalmente su clúster. Los clústeres elásticos gestionan prácticamente cualquier cantidad de lecturas y escrituras con petabytes de capacidad de almacenamiento mediante la distribución de los datos y computación entre varios volúmenes e instancias de procesamiento subyacentes. Para obtener más información, consulte Uso de clústeres elásticos de Amazon DocumentDB.

Historial de documentos de la Guía para desarrolladores de Amazon DocumentDB

- Versión de la API: 31-10-2014
- Última actualización de la documentación: 27 de mayo de 2026

En la siguiente tabla se describe la documentación de esta versión de la Guía para desarrolladores de Amazon DocumentDB.

Cambio	Descripción	Fecha
Nueva guía para desarrolladores de Amazon DocumentDB	La guía para desarrolladores de Amazon DocumentDB ya está publicada en una nueva URL. La referencia de la API está disponible por separado en Amazon DocumentDB API Reference.	27 de mayo de 2026
AWS actualización de política gestionada: cambio de política	Amazon DocumentDB actualiza las políticas de acceso completo para los clústeres elásticos.	11 de febrero de 2025
AWS actualización de política gestionada: cambio de política	Amazon DocumentDB actualiza las políticas de acceso completo para los clústeres elásticos.	21 de febrero de 2024
AWS actualización de política gestionada: cambio de política	Amazon DocumentDB actualiza las políticas de solo lectura y acceso completo para los clústeres elásticos.	21 de junio de 2023
AWS actualización de política gestionada: nueva política	Amazon DocumentDB presenta una nueva política	8 de junio de 2023

	de solo lectura para clústeres elásticos.	
AWS actualización de política gestionada: nueva política	Amazon DocumentDB presenta una nueva política de acceso completo para clústeres elásticos.	5 de junio de 2023
Compatibilidad con MongoDB 5.0	Amazon DocumentDB ahora es compatible con la versión 5.0 de MongoDB.	1 de marzo de 2023
actualización de la política	Para admitir la función de clúster elástico de Amazon DocumentDB, la AmazonDocDBConsoleFullAccess política se actualiza y AmazonDocDB-ElasticServiceRolePolicy se introduce.	30 de noviembre de 2022
Clusteres elásticos	Se ha añadido una nueva característica de los clústeres elásticos que admite la partición basada en hash (sharding) de datos del sistema de almacenam iento distribuido de Amazon DocumentDB.	30 de noviembre de 2022
Clústeres globales	Se agregó documentación sobre cómo usar los clústeres globales.	2 de junio de 2021
Suscripciones a eventos	Se agregó la documentación de suscripción a eventos.	26 de marzo de 2021

Actualizaciones a la versión 3.6	Mejoras documentadas de la versión 3.6 en los controles de acceso basados en roles, los operadores de agregación y el rendimiento.	15 de enero de 2021
Compatibilidad con MongoDB 4.0	Amazon DocumentDB ahora es compatible con la versión 4.0 de MongoDB.	9 de noviembre de 2020
Guías de introducción	Nuevas guías de introducción para empezar a utilizar Amazon DocumentDB con Amazon EC2 AWS Cloud9, Robo3T o Studio3T.	15 de agosto de 2020
Zonas de disponibilidad adicionales admitidas	Amazon DocumentDB agregó compatibilidad con una zona de disponibilidad adicional en Asia-Pacífico (Seúl) (ap-northeast-2).	14 de julio de 2020
Se ha agregado compatibilidad para la copia de instantáneas entre regiones.	Amazon DocumentDB ha añadido soporte para copiar instantáneas de clústeres entre Regiones de AWS. Para obtener más información, consulte Copiar instantáneas en las regiones.	10 de julio de 2020

Se ha añadido soporte para la clase de instancia T3.	Se agregó compatibilidad con los tipos de instancias T3 en todas las regiones que admiten Amazon DocumentDB. Para obtener más información, consulte Clases de instancias admitidas por región y Especificaciones de clases de instancias.	30 de junio de 2020
Se ha añadido soporte para. AWS GovCloud (US)	Amazon DocumentDB ya está disponible en la AWS GovCloud (US) región (us-gov-west-1).	29 de junio de 2020
Se agregaron 16 métricas nuevas. CloudWatch	Amazon DocumentDB agregó soporte para 16 CloudWatch métricas nuevas de Amazon. Para obtener más información, consulte Monitorización de Amazon DocumentDB con. CloudWatch	23 de junio de 2020
Se ha añadido compatibilidad para caracteres nulos y operador $regex.	Amazon DocumentDB ha añadido compatibilidad para caracteres nulos en cadenas y la capacidad de usar un índice para $regex. Para ver las API de MongoDB compatibles y las capacidades de canalización de agregación de Amazon DocumentDB, consulte Diferencias funcionales con MongoDB.	22 de junio de 2020

Se ha agregado compatibilidad para las capacidades mejoradas de indexación con varias claves.	Amazon DocumentDB ha agregado compatibilidad para las capacidades mejoradas de indexación con varias claves, incluidas la indexación de matrices superiores a 2048 bytes y la capacidad de crear un índice compuesto de varias claves con varias claves en la misma matriz. Para obtener más información, consulte Diferencias funcionales con MongoDB.	23 de abril de 2020
Se agregó soporte para la protección contra la eliminación de una pila de Amazon DocumentDB. CloudFormation	Amazon DocumentDB agregó soporte para habilitar la protección contra eliminaciones al crear una pila de Amazon DocumentDB. CloudFormation	20 de abril de 2020
Se ha añadido compatibilidad con control de acceso basado en roles.	Amazon DocumentDB ha añadido compatibilidad con control de acceso basado en roles mediante roles integrados.	26 de marzo de 2020
Se ha añadido compatibilidad para una zona de disponibilidad adicional en Canadá (centro) (ca-centro-1).	Amazon DocumentDB ya está disponible en la región Canadá (centro) (ca-centro-1) con instancias de clase R5 y tres zonas de disponibilidad.	26 de marzo de 2020

Se ha añadido compatibilidad para dos API de MongoDB adicionales.	Amazon DocumentDB añadió compatibilidad con las API de `$dateFromString` y MongoDB `executionStats` .	23 de marzo de 2020
Se ha agregado compatibilidad con cinco API de MongoDB adicionales.	Amazon DocumentDB añadió compatibilidad con las API de `$objectToArray` , `$arrayToObject` , `$slice`, `$mod` y MongoDB `$range`.	6 de febrero de 2020
Se ha añadido compatibilidad con Canadá (centro).	Amazon DocumentDB ya está disponible en la región Canadá (centro) (ca-centro-1) con instancias de clase R5.	11 de diciembre de 2019
Se agregó soporte para. ChangeStreamLogSize	Amazon DocumentDB ha añadido soporte para `ChangeStreamLogSize` para las métricas de Cloudwatch.	22 de noviembre de 2019
Se agregó soporte para la región Europa (París)	Amazon DocumentDB ya está disponible en la región de Europa (París) (eu-west-3) con instancias de clase R5.	30 de octubre de 2019
Se agregó soporte para la región Asia-Pacífico (Bombay)	Amazon DocumentDB ya está disponible en la región Asia-Pacífico (Bombay) (ap-south-1) con instancias de clase R5.	17 de octubre de 2019
Compatibilidad añadida con tres API de MongoDB adicionales	Amazon DocumentDB agregó compatibilidad con las API de `$addFields` , `$concatArrays` , de `$lookup` MongoDB.	16 de octubre de 2019

Se agregó soporte para la región Asia-Pacífico (Singapur)	Amazon DocumentDB ya está disponible en la región Asia-Pacífico (Singapur) (ap-southeast-1) con instancias de clase R5.	14 de octubre de 2019
Se añadió un nuevo documento para actualizar certificados TLS	Se añadieron instrucciones para actualizar los certificados CA de tal forma que se utilice el nuevo para crear las conexiones TLS.	2 de octubre de 2019
Se añadió soporte de API para los certificados	Amazon DocumentDB incorpora un nuevo tipo de certificado de datos para las instancias. Para obtener más información, consulte DBInstance.	1 de octubre de 2019
Compatibilidad con la creación de perfiles de consultas	Amazon DocumentDB ha añadido la posibilidad de crear un perfil de las operaciones admitidas en las instancias y bases de datos del clúster.	19 de agosto de 2019
Se agregó una tercera zona de disponibilidad en Asia-Pacífico (Tokio)	Amazon DocumentDB agregó una tercera zona de disponibilidad (AZ) para sus instancias informáticas en Asia Pacífico (Tokio).	9 de agosto de 2019

Compatibilidad con API de Mongo adicionales	Se agregó soporte para capacidades de canalización de agregación adicionales que incluyen los operadores `$in`, `$isoWeek`, `$isoWeekYear` , `$isoDayOfWeek` , así como operadores de agregación `$dateToString` y la etapa de agregación `$addToSet` . Amazon DocumentDB también agregó compatibilidad con el comando `top()` para el diagnóstico a nivel de colección y la posibilidad de modificar el parámetro `expireAfterSeconds` de los índices TTL mediante el comando `collMod()` .	31 de julio de 2019
Compatibilidad añadida para la región Europa (Londres)	Amazon DocumentDB ya está disponible en Europa (Londres) (eu-oeste-2) con instancias de clase R5.	18 de julio de 2019
Se han añadido muestras de código	Se han añadido ejemplos de código en R y Ruby para la conexión mediante programación a Amazon DocumentDB.	17 de julio de 2019
Se ha añadido una práctica recomendada	Se ha añadido una práctica recomendada para ayudarle a administrar sus costos de Amazon DocumentDB.	17 de julio de 2019

Compatibilidad con la detención e inicio de un clúster	Amazon DocumentDB ha incluido compatibilidad con la detección e inicio de clústeres para administrar los costos de entornos de desarrollo y pruebas.	1 de julio de 2019
Compatibilidad con la protección contra eliminación del clúster	Amazon DocumentDB ha incluido la protección contra eliminación para proteger sus clústeres contra la eliminaci ón accidental. Para obtener más información, consulte los siguientes temas: Creación de un clúster de Amazon DocumentDB, Modificación de un clúster de Amazon DocumentDB, Eliminación de un clúster de Amazon DocumentDB, y `DeletionP rotection` en el tema de la API DBCluster.	1 de julio de 2019
Actualización de diferencias funcionales	Se han añadido transacciones implícitas a las diferencias funcionales.	26 de junio de 2019
Adición de diferencias funcionales	Se ha añadido una nota en relación con el almacenam iento y la compresión del índice en Amazon DocumentD B.	13 de junio de 2019
Se admite una región adicional	Amazon DocumentDB ya está disponible en la Asia-Pacífico (Sídney) (ap-sureste-2) con instancias de clase R5.	5 de junio de 2019

La clase de instancia R5 se admite en otras regiones	Se agregó compatibilidad con la clase de instancia R5 para 4 regiones adicionales: EE.UU. Este (Ohio), EE.UU. Este (Norte de Virginia), EE.UU. Oeste (Oregón) y UE (Irlanda). Con este cambio, las instancias R5 son compatibles en todas las regiones compatibles con Amazon DocumentDB.	17 de mayo de 2019
Se admiten regiones adicionales	Se agregó soporte para 2 regiones adicionales, Asia Pacífico (Tokio) (ap-northeast-1) y Asia Pacífico (Seúl) (ap-northeast-2) con clases de instancias R5. Para obtener más información, consulte Clases de instancias admitidas por región y Especificaciones de clases de instancias.	8 de mayo de 2019
Se han añadido más ejemplos del código de conexión	Se han añadido ejemplos de código en Java y C# para conectarse a Amazon DocumentDB.	24 de abril de 2019

Soporte adicional para la API de Mongo	Se ha añadido compatibilidad adicional con siete operadores de cadenas de agregación (`$indexOfBytes`, `$indexOfCP`, `$strLenBytes`, `$strLenCP`, `$toLower`, `$toUpper`, y `$split`), nueve operadores de fecha y hora (`$dayOfYear`, `$dayOfMonth`, `$dayOfWeek`, `$year`, `$month`, `$hour`, `$minute`, `$second`, y `$millisecond`) y la etapa de canalización de agregación `$sample`.	4 de abril de 2019
Se han añadido ejemplos del código de conexión	Se agregaron ejemplos de código en Python, PHP y Go para conectarse a Amazon DocumentDB Node.js.	21 de marzo de 2019
Compatibilidad con la región de Fráncfort e instancias R5	Se ha añadido compatibilidad con la región Europa (Fráncfort) (eu-centro-1) con clases de instancias R5. Para obtener más información, consulte Clases de instancias admitidas por región y Especificaciones de clases de instancias.	13 de marzo de 2019

Compatibilidad con operadores de canalización de agregación	Se ha añadido compatibilidad con nuevos operadores de cadenas de agregación (`$concat`, `$substr`, `$substrBytes`, `$substrCP`, `$strcasecmp`), un operador de agregación de matriz (`$size`), un operador de acumulador de grupo de agregación (`$push`) y etapas de agregación (`$redact` y `$indexStats`). También se ha añadido compatibilidad con operadores de matriz posicional (`$[]` y `$[<identifier>]`) y `hint()`.	28 de febrero de 2019
Actualizaciones del motor	Se ha añadido documentación para determinar las modificaciones del clúster pendientes y para actualizar la versión del motor del clúster.	15 de febrero de 2019
Auditoría de eventos	Se agregó soporte para auditar eventos de bases de datos con Amazon CloudWatch Logs.	12 de febrero de 2019
Quick Start (Inicio rápido)	Se ha añadido un tema de inicio rápido para ayudarle a empezar a utilizar CloudFormation Amazon DocumentDB de forma sencilla.	11 de enero de 2019

Versión pública	Esta es la versión pública inicial de Amazon DocumentDB (compatible con MongoDB). Esta versión incluye la Guía para desarrolladores y la Referencia de la API para administración de recursos.	9 de enero de 2019

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.